DEVELOPMENT OF THE PRESTO PEN-BASED MUSIC EDITOR

A THESIS

SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE

OF

MASTER OF SCIENCE IN COMPUTER SCIENCE

IN THE

UNIVERSITY OF CANTERBURY

by

Elizabeth P Y Ng

University of Canterbury

1998

Abstract

Deficiencies in existing methods of music input have made the process slow and tedious; text music editors are difficult to use, graphical music editors are slow, and Optical Music Recognition is inaccurate and requires cumbersome equipment. This research aims to develop one of the fastest and easiest methods for entering music into computers—the pen-based music input.

The design of pen music input closely follows the idea of traditional pencil and manuscript for musicians. Musicians draw sketches on the system similar to those they do on paper. The development of smaller and lighter low-cost pen computers enables the system to be portable.

This thesis develops an earlier method for using pen computers to input music, called *Presto*. It extends the gesture set and further develops the system in Presto. The objectives of Presto are to be fast, easy to learn, easy to use, and portable. This thesis first reviews existing gesture sets and graphical music editors. Next, four aspects of Presto are investigated: gestures, editing features, drawing beams, and feedback. The research on each part reviews existing work on the relevant issues, then designs and implements improvements. Presto has become significantly faster than its previous version, and many features are improved to make it more usable. It can be three to more than four times faster than using other methods to input music.

The research on the pen-based music input has greatly improved accuracy, functionality, speed, and usability of Presto, making it a useful technique for musicians.

Acknowledgements

First, I thank deeply my supervisor, Dr Tim Bell, for his endless assistance, guidance, and advice throughout this research, reading through every draft that I have given him. I also thank Dr Andy Cockburn for helping and correcting me on the human-computer interaction areas, especially during the period when Tim was on study leave. I am also grateful to him for reading all the chapters in this thesis.

Thanks to Mr Martin Setchell from the School of Music for his input on the musical aspects. Many thanks also to Dr Roger Buckton, the Head of Department of the School of Music, Mr Chris Cree Brown also from the School of Music, and the musicians who participated in the experiments.

Thanks to Professor Harold Thimbleby, Faculty Director of Research and Professor of Computing Research of the Department of Computer Science in Middlesex University, and Mr Thomas Heck for their ideas and encouragement when they tried the Presto system.

I am grateful to the technical and support staff in the department for their help and support during this research. Many thanks also to the librarians in the university library for their time and effort to help me find research materials. Finally, I thank Mr Stuart Yeates for proofreading and commenting on the final draft of this thesis, Mr Jamie Anstice for supporting the pen computing aspects, and Mr Brandon Harris for sharing the joys and sorrows in postgraduate life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In 1995, the Association for Computing Machinery (ACM) presented an *Interactions* Design Award to the *muse* [grae96$_a$, grae96$_b$], an electronic music stand designed for orchestral musicians. The muse consists of a portable digital display and a matching stand illustrated in Figure 1.1; the display attaches to the stand whenever it is needed in rehearsals and performances. When the musician writes on the foldable and touch-sensitive display, the muse translates the writing into a command or graphical representation. In this way, the musician marks up music using only the display and the pen. Such pen input method is likely to be popular in music systems, and the muse is only one example of how pen music input can be used. However, drawing the detailed features of symbols, such as filling in noteheads or drawing note stems, may be redundant and time-consuming, thus this research aims to design a method using the pen to draw efficient gestures (or shorthand) to input music into the computer.

1.1 Aim

Every method to enter music into a computer has its benefits and deficiencies. In early systems, the only method was to type music representation languages into a text editor. This method does not require special equipment or musical skill, but is tedious and cumbersome to use because the keyboard and the screen were designed for text input and display rather than music. Some languages, such as MusicTex [taup95], are awkward to use, since their typesetting problems must be solved by the user. The languages are also difficult for novices, because the grammer is strict and not intuitive.

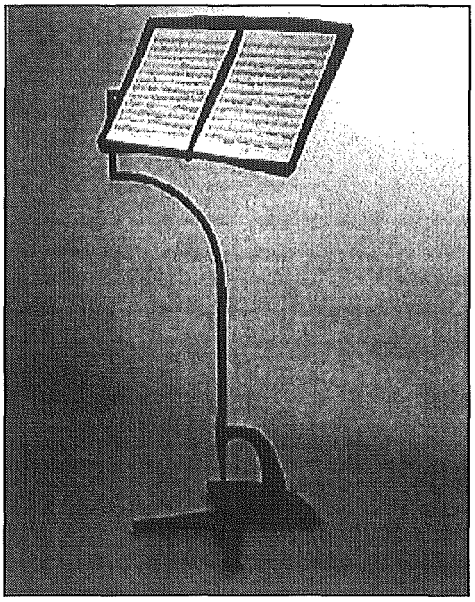


Figure 1.1: The *muse* ([grae96$_a$], p. 34).

Current music entry and editing packages usually allow input from the keyboard, the mouse or other pointing devices. In these systems, the pitch and the duration has to be selected separately and consecutively. Other systems also allow users with musical skills to input music by playing on a Musical Instrument Digital Interface (MIDI) instrument connected to the computer. However, if there are multiple parts on a staff or if the music is an orchestral piece, the user might find it easier to enter each line of staff separately. This increases the time required to enter music and introduces the problem that the rhythm played for each line may not be completely accurate. Another problem is that it is difficult to enter enharmonic equivalents, where two notes are written differently but played the same way.

Graphical User Interfaces (GUIs) allow music entry on a bit-mapped display with the keyboard, a pointing device which is typically a mouse, and maybe a MIDI instrument. This method gives immediate feedback and is easier to use than the text entry method, because it presents information graphically. However, this is slow because mouse input is not natural. Moreover, it demands more computing power than other methods and users may experience delay when they use such a system.

Soundtrack analysis is the fully computerised transcription from a recorded musical work. This method requires a recording to be available and has many problems

associated with spectral analysis, acoustics, and input via synthesizer keyboard that are not yet solved [mcna96$_a$, mcna96$_b$, pisz77, pisz79].

The process of Optical Music Recognition (OMR) involves scanning printed sheets of music and translating the graphical objects in the scanned image into musical features [bain96, bain97]. This method allows whole pages of a score to be entered at once, but the score must be in reasonably good condition and a scanner is required. OMR can input textual information such as lyrics and dynamic markings, but not all musical features can be recognized and post-OMR editing is needed. Even if OMR is 99% accurate, the score still needs to be proofread to find the 1% errors. Thus, score input for OMR is relatively fast, while error correction is much slower.

In general, these methods to input music into the computer are either slow, inaccurate, tedious, or they need some special hardware. A detailed analysis can be found in Anstice [anst96$_a$].

An alternative method to enter and edit music on-line and graphically is by using a pen-based computer (or simply, a pen computer). A pen computer is a system in which the user writes on the computer screen with an electronic writing device called a pen or a stylus, and the system immediately processes the pen movements and updates the screen. The pen not only acts as a pointing device but also inputs commands and data through gestures. A gesture is a combination of strokes or marks produced electronically by the pen.

The design of a pen computer is based closely on the idea of traditional pen and paper (or pencil and manuscript in the case of a musician), and the developments of smaller and lighter pen computers enable the system to be more portable. To write music on this system, music writers can draw similar sketches to those they do on paper. However, the writer can also print, publish, and distribute the score in a format that is generated by the computer and understood easily by other musicians; shorthand notations on paper will remain as sketches, which are only comprehended by trained readers. Writing music into the computer also allows faster and easier publication than writing onto paper.

Anstice [anst96$_a$] proposed, designed, and implemented a music editor using the pen input method. He designed a set of shorthand pen gestures and developed a prototype music system that uses these gestures. He then tested the system and found that this method can be three times as fast as other methods of music input. This thesis extends

the gesture set and further develops various parts of the music system. The system is intended for composers, arrangers, music editors, and performers.

1.2 Music printing

In the early 1960s, the music printing industry became interested in the power of computing, and the potential of computerized music printing had been considered by dal Molin in 1953. The terminals that dal Molin designed are presented in Section 2.2.1. At that time, many computer-literate musicians found it challenging to apply their computing knowledge to musical notation, which is a combination of "language on the one hand and mathematics on the other." [krum90]

Printing, including music printing, is "a technique for producing many identical copies taken from raised, incised or plane surfaces." [krum90] Before the technique was established and used widely, music was preserved and distributed in manuscript or in the oral collection among priests and musicians.

In the 1480s, woodcut music was printed from woodblocks or metalblocks, but printing results were not satisfactory, especially if the printers were not careful. Then in the 1500s, printers started to print music from type. The punchcutter and the founder had to be very skilful to get quality printed results. There were problems if the printers used types that were badly fitted, indifferently cast, and with only a limited set of characters. This method was also clumsy and could not show more than one part on a staff.

Engraving started in the 1530s when hand-drawn lines were engraved in copper. However, this method was slow and expensive. In the 1790s, lithography was invented. It was the process of writing or drawing a design on a special stone called the *lithographic stone*, and impressions of it in ink could be taken. This process was an advancement in music printing but it was still considered slow.

Music typewriters, like the one in Figure 1.2, were developed from the 1830s to the end of the century. Computers came into play in music printing in the 1940s. Later in the 1970s, several projects on computerized music systems were started. However, it was only in 1981 that the International Business Machines Personal Computer (IBM PC) and the Apple Macintosh made computers more accessible for researchers to develop faster and less expensive music systems [gomb77, cart88, trau91, byrd94]. Some of these music systems are presented in Section 2.2 and Section 2.3. The details of the history of

music printing are found in Krummel and Sadie [krum90]. Ross [ross70] and Read [read69] describe the techniques in music printing.

Musicians usually write and edit music on paper by hand, which is time-consuming and tedious; entering music into the computer saves them a lot of time and effort. For example, musicians using a computer to write music do not need to have good handwriting. They can also produce and distribute computer-generated scores to other musicians; these scores are easier to read than handwritten ones. Manual tasks, such as creating parts separately, can be done automatically by extraction; without a computer, the musician has to rewrite each part by hand. In addition, musicians can manipulate the score in the computer by using editing features, such as cut and paste functions found in text editors, and music features such as automated transposition. Minor changes in a score can be made without rewriting a large portion. The playback features in the systems can also be used to check for errors in copying and allow composers to hear their compositions. In addition, musicians can analyse music in these systems.

This research limits its class of music to western Common Music Notation (CMN). This thesis also assumes that the reader knows CMN and musical terminology. Those who are not familiar with music can refer to Heussenstamm [heus87], Rader [rade96], and Bob [bobe97] for the definition of key musical terms, and Taylor [tayl92, tayl94] for a basic guide to CMN.

Figure 1.2: The *musicwriter* designed by Effinger in 1973 ([krum90], p. 64).

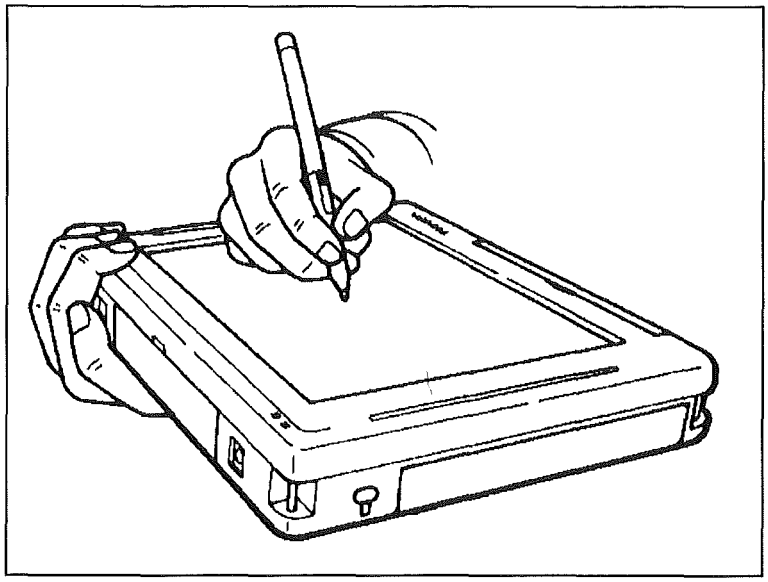


Figure 1.3: An example of a pen-based computer.

1.3 Pen-based computing

The main objective of pen-based computing is to closely resemble the pen and paper interface [meye95, meye97]. It aims to help users enter data in a more natural way, and to lower the barrier between computer-illiterate users and computers [klau93].

The pen computer brings us a step closer to the realisation of the *Dynabook*, a concept that Alan Kay first described in 1968 [ryan$_1$91]. The Dynabook is the size of a notebook wirelessly connected to the network, and can be used immediately by novices. Although the Dynabook still does not exist, the concept of pen computing is becoming increasingly important in the fields of computer communication and human-computer interaction [mill$_1$93].

A pen computer has a tablet, a pen, and software that recognizes and converts handwritten text to printed text [hela88, mack95, meye95, meye97]. A tablet is a flat monitor that records and display marks from the pen. An example of a pen computer is in Figure 1.3. Meyer [meye95, meye97] describes the history and an overview of pen computing in detail.

1.3.1 Benefits

People can use a pen more efficiently than a mouse, because pens are more intuitive, usable and natural [babe97]. The pen is also more useful than a mouse for tasks requiring fine motor coordination, such as freehand drawing, or for tasks that concentrate on creating contents, for example preparing a first draft or composing music [tapp90].

A keyboard is generally preferable for entering text into computers, but there are significant advantages in using a pen to input non-standard text, such as mathematical formulae or musical notation [fran95]. The pen is also faster and more efficient than a mouse or keyboard for annotating text [babe97]. Moreover, a keyboard is used to input text, and a mouse is used for positioning text and selecting menu commands, but the pen itself could enter text, define the size and location of text, as well as draw graphic elements at the same time [meye95, meye97]. For these tasks, the characteristics of the pen let the user interact faster and more fluently than other input methods.

A good pen interface should not have different modes for recognizing handwriting; this will be easier for users, because non-modal systems would not have mode errors. The cursor from conventional text-based and graphical user interfaces for marking the position of input is not required because human-computer interaction is more direct [meye95, meye97]. Moreover, pen computers can be hand-held, thus they are portable and do not require a stable work surface [fran95].

Several experiments with pen computers have shown that users are positive about pen computing and would be willing to use the pen-based interface in addition to the keyboard [brig$_2$92, fran95, over96].

1.3.2 Drawback

The drawback of pen-based systems is the poor handwriting recognition available; the recognition is slow and inaccurate [fran92, chan$_2$94, leed94, mack94, mack97$_a$]. In addition, a user's handwriting is personal and unique, training a computer to recognize one's handwriting would mean that another user has to train the system again before using it. Thus, despite of the benefits of the pen computer, many users are still unwilling to use the pen-based interface instead of the keyboard [brig$_2$92]. Tappert et al. [tapp90] and Tappert [tapp91] describe the state of the art of handwriting recognition.

1.3.3 Gestures: An alternative to handwriting recognition

One solution to the problem of handwriting recognition is not to deal with the recognition itself, but to force the user to write in a predefined way which allows only little variation [meye95, meye97]; these predefined strokes are called gestures [wolf86, wolf87, henr90]. An example of a gesture set is Unistrokes which is presented in Section 2.1.1. A gesture set is a group of gestures designed for use in a program.

Although predefining gestures will restrict users and force them to learn and remember a new set of writing, the two major benefits of gestures outweigh the requirements: accuracy and speed.

A system can recognize gestures more easily and accurately than undefined handwriting, if the gestures are designed to be simple yet different and simple recognition methods are used [tapp90]. Users may also find simple gestures, that are also mnemonic to their equivalent output, easy to learn.

Users can draw gestures to input commands and data into a system faster than using a keyboard or a pointing device, because gestures do not require two actions like conventional input does. A command issued by a keyboard or a mouse usually requires at least two actions: the command to be performed and the position to perform the command. On the other hand, a single gesture indicates both the command to be performed, and the object of that command [mezi93, fran95]. The command selection is determined by the shape of the gesture, and the object selection by the position of the gesture at either the beginning (which is more common) or the end. Careful selection of the gestures used can result in quicker input than other input systems. For instance, more common symbols are drawn in shorter strokes than less common symbols. In addition, the user would achieve greater efficiency in task performance [meye95, meye97]. Gesture input also increases the user's feeling of control over the system, because gestures are drawn right on the object [mill$_1$93].

1.4 Pen-based music input: An emerging concept

The advantages of pen-based computing would benefit music data entry. A pen computer imitates the traditional manuscript that musicians have been using, and they could accept and learn the new system quickly. Existing pen music systems are the

CNMAT system [lero94] and the Pitchforth's system [pitc94] that are presented in Section 2.3.4 and Section 2.3.5 respectively.

However, if the problem of handwriting recognition has made entering text into the pen computer difficult, the task of entering music would also be tedious. For example, the CNMAT editor has to guess from the user's sketches what the intended musical symbols are; the results may not be what the user expects. The user of the Pitchforth's system have to write several times before the system is trained to recognize the individual's handwriting. Moreover, a user's handwriting is personal and unique, this means that another person has to train the system again before using it.

Using gestures is an ideal solution to the problem of handwriting recognition for music input. An example of a gesture-driven music system is Char-rec [buxt79, buxt86$_a$] which is presented in Section 2.1.5.

1.5 Presto gesture set and music editor

The current research on using pen computers to input music was started by Anstice [anst96$_a$]. He first conducted an experiment to investigate how musicians write music and used the observations to design the Presto gesture set. The design of this gesture set was also influenced by Unistrokes, which is a character gesture set presented in Section 2.1.1. Anstice's design depended on the frequency of musical symbols; the more frequent the symbol occurs the shorter the gesture. Anstice implemented the gesture set in Presto and evaluated on the input speed of the system. He found that handwriting music on paper is about one-third of the time for other input methods, and that pen input could be as fast as or faster than handwriting; this means that pen input would be at least three times faster than other input methods [anst96$_a$]. In this thesis, the research model including the gesture set and the system will be called *Presto*, the gesture set that Anstice designed will be called *Presto1*, and the system that was developed to test Presto1 will be called *MusEd1*.

This thesis further develops the concept of Presto towards these objectives:

- The input into the system must be fast. This aim is achieved by using more efficient entry methods, for example using fast and short gestures to write common musical symbols (which is also the aim of Unistrokes), and automating repetitive or redundant tasks, such as drawing ledger lines, stems, and filling noteheads.

- All gestures must be short and consist of single strokes. Short gestures enable fast input, and single strokes remove the problem of reconstructing more than one stroke into a single glyph in the process of recognition.
- All gestures must be easy to draw, so that there is little variation in the gestures drawn for the same symbol or command and the system could recognize them easily. In addition, users could recall the gestures easily.
- All gestures must be easy to remember and mnemonically linked to their equivalent musical symbols.
- The system must be portable to allow the user to use it anywhere.
- The system itself must be easy to learn, so that the gap between a musician and the computer can be narrowed.

This research is divided into four parts: Presto gestures, editing features, drawing beams and feedback. These parts are studied separately and refined to better achieve the objectives of Presto. The study on each part includes a review of existing work on the relevant issues, the design, and the implementation of these issues in the system. Results from evaluations show that musicians found the system useful and the speed of entering music in Presto could be three to more than four times faster than using other methods to enter music into the computer. *Presto2* in this thesis refers to the gesture set that the author developed from Presto1, and *MusEd2* refers to the system that is developed from MusEd1 to test Presto2.

1.6 Synopsis

Chapter 2 in this thesis reviews five gesture sets, four graphical music editors and five pen-based music editors. It also highlights the positive features in these systems that Presto could acquire and the negative points that Presto should avoid.

Chapter 3 to Chapter 6 discusses the four main parts of the research. Each chapter first reviews the existing concepts and prior work on the related issues, designs these parts in Presto2, and presents the methods of implementating them into MusEd2. Chapter 3 redesigns and redefines the gestures that users have problems with in Presto1. Chapter 4 discusses and designs some important editing features in Presto. Chapter 5 studies the problem of drawing beams in Presto1. Chapter 6 discusses some feedback issues and

adds them to Presto; these include visual and audio feedback from the system when the user inserts, deletes or changes the properties of a musical symbol in Presto, and help and error messages in the system.

The evaluations on the usefulness and the speed of Presto is presented in Chapter 7. Future work of the relevant issues are also discussed. Finally, Chapter 8 concludes the thesis by listing the contributions that Presto gave to the field of computer music.

There are six appendices at the end of the thesis: all the gestures in Presto2 are in Appendix A, a reference manual to MusEd2 is in Appendix B, an example of the questionnaire in the beam survey that is presented in Chapter 3 is in Appendix C, the model answers to the beam survey questionnaire is in Appendix D, an example of the questionnaire in the evaluation that is discussed in Chapter 7 is in Appendix E, and an analysis of the results of the evaluation questionnaire is in Appendix F.

Chapter 2

Gestures Sets and Music Editors

This chapter reviews five gesture sets and their design principles. Four graphical music editors and five pen-based music editors are also discussed, focusing on the objectives of each editor. These gesture sets and editors have their individual positive points that could be incorporated into Presto, and weak points that Presto should avoid.

2.1 Gesture sets

This section reviews five gesture sets: Unistrokes, Graffiti, Gesture Mosaic, Pitman, and Char-rec. A gesture is a combination of strokes that produces actions. A gesture set is a group of gestures that are designed to use in a program.

Unistrokes, Graffiti, and Gesture Mosaic use gestures to represent the Roman alphabet, Pitman uses gestures to represent the phonetic sound of the English language, and Char-rec uses gestures to represent musical symbols. QuickPrint is a commercial handwriting recognizer and trainer that does not have a gesture set, and it is mentioned briefly in this section.

2.1.1 Unistrokes

Unistrokes is a special set of gestures developed by Goldberg and Richardson [gold93] at Xerox Palo Alto Research Center (PARC). It is simplified from the Roman alphabet and has a goal similar to that of touch-typing; that is, to enter text *eyes free* in activities such as copying and group meetings. There are three main principles for the design of Unistrokes: it must be easy to learn, it must be well separated in sloppiness in space, and it must be fast to write.

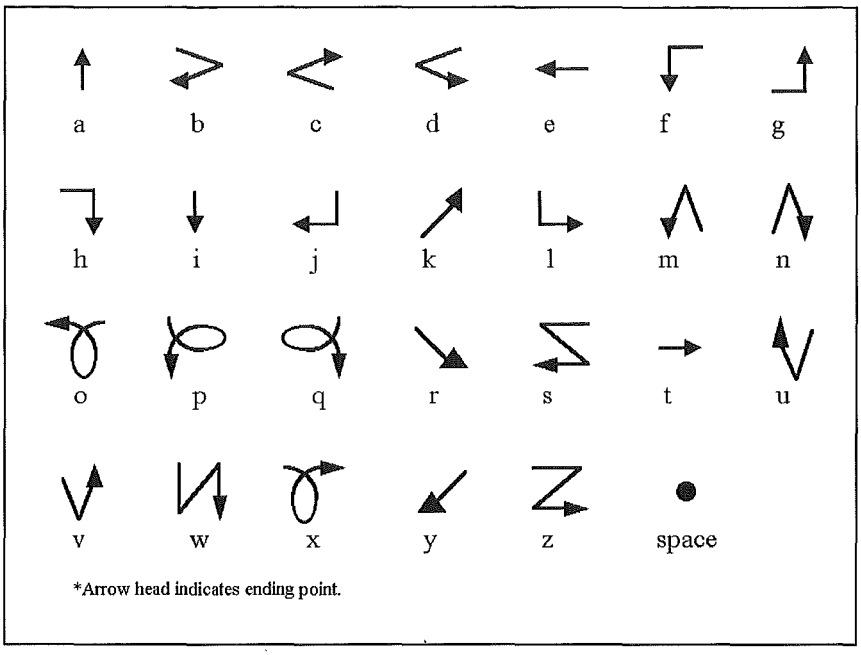


Figure 2.1: Unistrokes gesture set [unis97].

Unistrokes has five major gestures that can be drawn in different directions and rotations. Each gesture is drawn with a single stroke (thus the name *Uni*strokes) and is assigned to represent a Roman letter according to the letter's frequency of use. The more often a letter is used, the smaller its assigned gesture is. For example, the most frequently used character, *space*, is assigned a dot gesture and the more common letters (*e*, *a*, *t*, *l*, and *r*) are assigned straight line gestures. Figure 2.1 shows the whole set of Unistrokes gestures. Each gesture is unique and it symbolises its corresponding letter. For example, the gesture assigned for the letter *c* symbolises the curvature of the letter, and the gesture assigned for the letter *e* symbolises the horizontal line in the middle of that letter.

Since every gesture consists of only one stroke, the user writes each gesture by starting at one point of the gesture and continuing through to the other point in the direction of the arrow head in Figure 2.1. Recognition of the gesture starts after the user lifts the pen. The user has to press a button on the barrel of the pen to draw upper-case letters with the same gestures that are for the lower-case letters.

Goldberg and Richardson tested Unistrokes with a mail-sending program. They recorded the time from pen-down, when the user starts to draw, to pen-up, when the user stops, and the interval between each gesture drawn. After they analysed the data, they found

that the delay between strokes is a critical influence in writing speed, a point already noted in shorthand books. Goldberg and Richardson also found that the movement distance from one stroke to another, for example to end a stroke low and start another stroke high, does not correlate with the movement time and the distance between two strokes does not affect the writing speed.

All six users learned Unistrokes with a training programme and all of them could write the gestures correctly without looking at a reference card after ten minutes. However, only three users used the mail-sending program regularly because there were only three pen tablets available. After the test, Goldberg and Richardson claimed that Unistrokes writing may support up to 3.5 letters per second, a rate that is slower than touch-typing (6 to 7 letters per second) by only half. Moreover, they found that strokes written from right to left are faster than strokes written from left to right, and strokes from bottom to top are slower than strokes from top to bottom. After users have tested Unistrokes, they said that they expected Unistrokes to be hard to learn but found it easy because of its similarity to the Roman alphabet. Thus, Goldberg and Richardson suggested introducing Unistrokes to users as a re-design of the Roman alphabet rather than as a new alphabet.

There are several advantages of Unistrokes over normal handwriting. Each gesture can still be distinguished from other gestures when it is written sloppily. There is only one stroke from pen-down to pen-up, thus segmentation errors are avoided. Segmentation errors occur if a gesture is composed of more than one stroke, and the system fails to differentiate correctly between the strokes. One stroke in a gesture also means that gestures can be written one on top of another unambiguously. This saves the writing space and movements of the hand. The most common letters are assigned to shorter gestures (straight lines) and a *space* character is assigned a dot gesture to speed up writing. However, even the longest gestures remain easy and fast to draw. Eyes-free operation for taking notes is possible.

Unistrokes has greatly influenced the design of Presto1, especially the rule that the most common characters be assigned to the shortest gestures. In Presto1, the crotchet is most commonly used and it is assigned the dot gesture [anst96_a, anst96_b].

2.1.2 Graffiti

Palm Computing Division of US Robotics developed Graffiti in 1994 [stew94, graf96, leap97, prin97]. Graffiti is a handwriting software that recognizes the Graffiti gesture set for entering text in pen-based systems such as Personal Digital Assistants (PDAs).

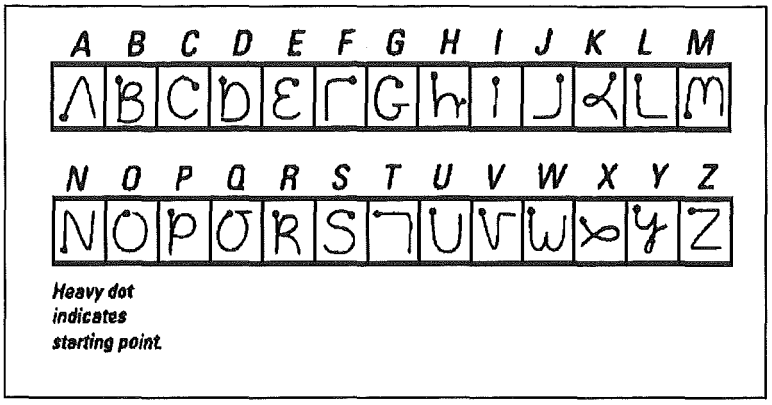


Figure 2.2: Graffiti character set [hewl+95].

Like Unistrokes in Section 2.1.1, the Graffiti gesture set, shown in Figure 2.2, is also a new set of alphabet simplified from the Roman alphabet, but the gestures in the Graffiti set resemble the upper-case letters [crot95]. In addition, there are other gestures that represent keys on the keyboard (such as *space*, *delete*, and *shift*) and actions such as copy and paste. Each gesture consists of a single continuous stroke and is clearly different from other gestures. In the system, the user draws a Graffiti gesture by beginning at the start point, represented by the heavy dot in Figure 2.2, and continuing until the end of the stroke. After the user lifts the pen, the system starts to recognize the gesture.

Palm Computing claimed that Graffiti is near 100 percent accurate and is at the rate of 30 words per minute [stew94]. Although users have to learn this new set of alphabet, Palm Computing claimed that the alphabet is easy to learn and easy to remember [leey94]. However, Crotty [crot95] advised users that they should expect a few weeks of practice before they get used to the system. Graffiti includes a system layout specially made for left-handed users.

Graffiti has many of the advantages that Unistrokes has over handwriting. A gesture can be recognized when it is written sloppily. There is only one stroke in each gesture thus avoiding segmentation errors. However, Graffiti mimics the Roman alphabet closer than Unistrokes, which means that Graffiti gestures may be longer than those of Unistrokes. This implies that Graffiti gestures are easier to remember but are slower to draw than Unistrokes gestures.

MacKenzie and Zhang [mack97$_b$] undertook the first empirical test to find out Graffiti's direct usability. They found that the accuracy rate was 97 percent after users have practised for five minutes. MacKenzie and Zhang also identified some of the letters (*x*, *k*, *u*, *v*, *f*, and *t*) which were more difficult and which affected the accuracy level. Wheelwright [whee96] described Graffiti as the "second wave" of pen-based technology, reviving the handwriting-recognition industry. He noted that teaching users a new set of characters is easier than developing a computer that will recognize all styles of handwriting.

2.1.3 Gesture Mosaic

Gesture Mosaic is a pen-based software character generation method developed by Mosaic Input Technologies, Division of Amtelco [mosa96]. The method is a fast and accurate way to enter Roman letters and numbers on PDAs such as the Apple Newton [mosa95$_a$, mosa95$_d$].

Gesture Mosaic provides a digital template or *mosaic* of character segments in the shape of a figure *8*, shown in Figure 2.3. The *gesture guides*, which are the thin lines in the figure, join the segments on the mosaic and specify the path of every gesture. The user enters the gestures across these character segments by following the gesture guide in the right direction. It is important that the user takes note of the segment that the gesture begins in and the segment that the gesture ends in.

Each gesture, consisting of one stroke, represents a unique characteristic of the shape of an upper-case letter, as illustrated in Figure 2.4, thus it is easy to learn and easy to remember [mosa95$_a$, mosa94]. Moreover, the fact that the stroke is always smaller than the actual letter results in the high speed of thirty to forty words per minute. A number lock button must be pressed first to enter numbers.

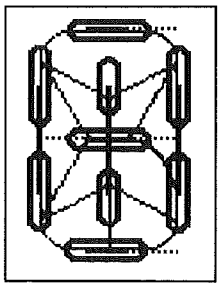


Figure 2.3: Gesture Mosaic template [mosa95$_d$].

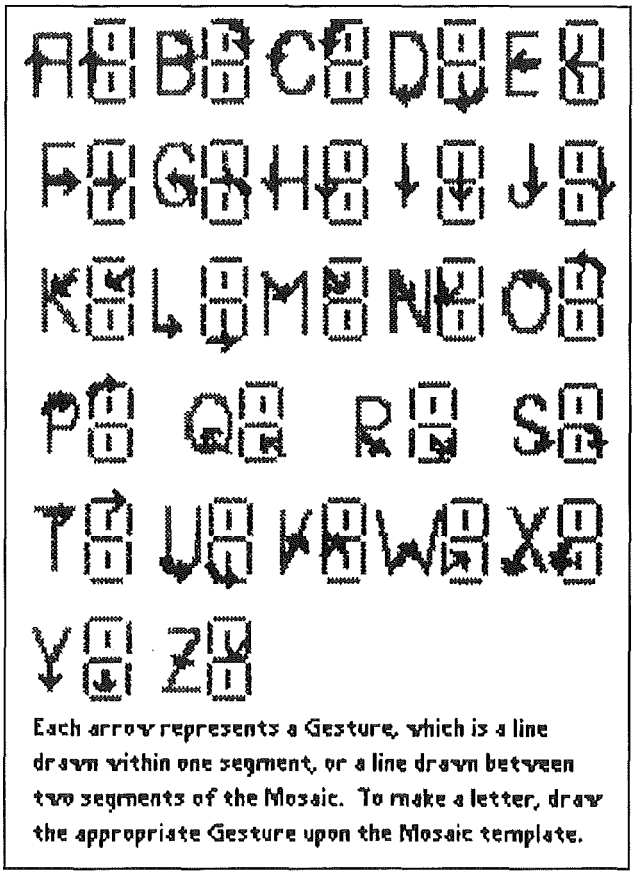


Figure 2.4: Gesture Mosaic gesture set [mosa97].

Gesture Mosaic contains recognition algorithms which can often interpret imperfect gestures. This is helpful when the user writes quickly and the gestures become sloppy. Letters and numbers have their own distinct sounds when they are recognized [mosa95$_a$]. Gesture Mosaic also uses less memory than similar programs, and a minimum amount of screen space [mosa95$_b$]. A user can thus create letters with near 100 percent accuracy, and overcome the problem of slow and inaccurate handwriting recognition.

Gesture Mosaic may be faster in writing speed than Unistrokes or Graffiti because very short strokes are assigned to every letter, but it may not be easier to learn than Graffiti since Graffiti symbolises the letters closer than Gesture Mosaic. Some users did find Gesture Mosaic hard to learn after using Graffiti for a few months, achieving only moderate results after more than an hour of use [mosa95$_c$]. An average user may take several weeks of daily use to become competent in using Gesture Mosaic.

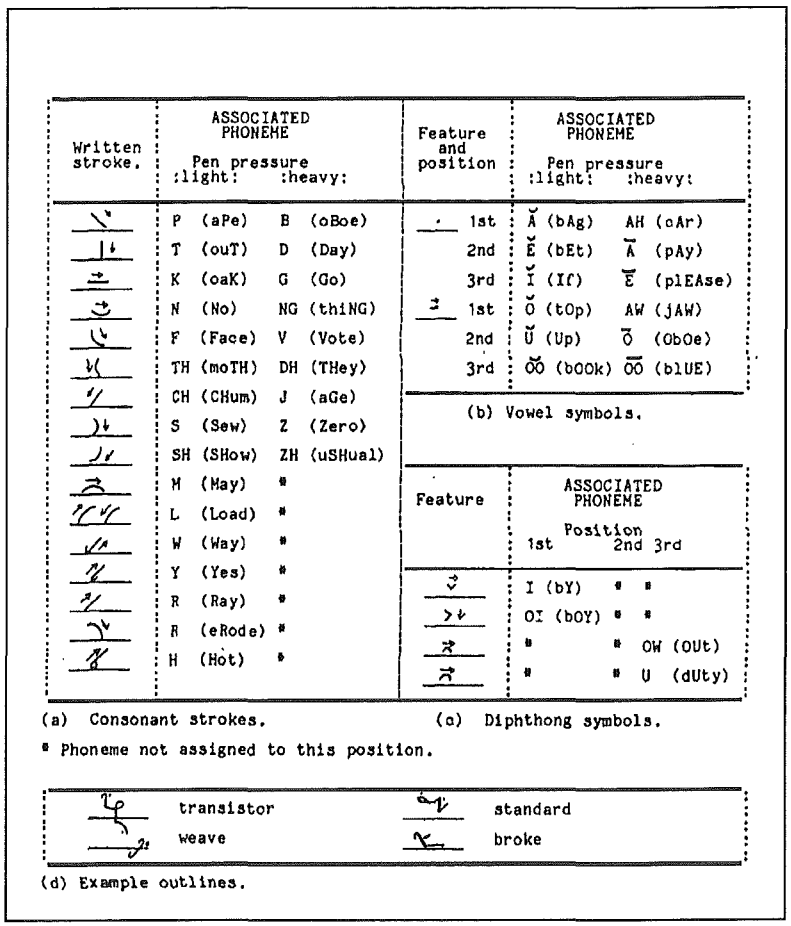


Figure 2.5: Pitman shorthand notation set ([leed84], p. 146).

2.1.4 Pitman

Pitman is a handwritten shorthand notation for pen and paper. It is used to record speech phonetically, thus it is very useful for transcription of speech in applications such as verbatim report and office dictation. Speech can be recorded at speeds of up to 120 words per minute using Pitman notation.

Pitman notation, shown in Figure 2.5, is constructed from a small number of simple shapes. However, there is no similarity between the letters and the shapes because the shapes corresponds to a phoneme of speech. There is a cognitive change in the minds of users when they use Pitman notation. They initially followed phonetic sounds, but they eventually jogged their memory. This seems to make the phonetic part of the process irrelevant, and the users' memory of the corresponding shapes becomes more important.

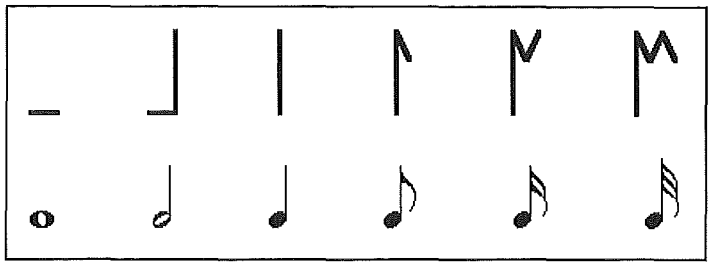


Figure 2.6: Char-rec gesture set ([buxt79], p. 21).

Leedham et al. [leed84, leed86] use Pitman notation to enter speech directly into a computer system which converts the gestures back to their phoneme equivalents. They found that visual or audio feedback about the gestures is important. If no feedback is given, the gestures will become sloppy and unrecognizable by others, including the computer system. The research also concentrates on the design of the pen used to draw the gestures on the system, and on the recognition of the gestures. Pitman notation has the advantage that it is the dominant form of shorthand [leed86] and many users may already know the notation, thus these users need not learn the notation.

2.1.5 Char-rec

The Structured Sound Synthesis Project (SSSP) at the University of Toronto has developed a set of tools, which is presented in Section 2.3.2, to help composers edit musical scores through the use of computer graphics [buxt79, buxt86$_a$]. One of the techniques to input notes in the SSSP editor is a set of note gestures called Char-rec (acronym from *char*acter *rec*ognizer).

Each gesture in Char-rec in Figure 2.6 represents a note of a certain duration, ranging from a semibreve to a demisemiquaver. Although every gesture consists of only one stroke, each one differs from another by its number of changes in direction. A user draws a note at a desired pitch by starting the gesture at that pitch and continuing until the end of the gesture, as illustrated in Figure 2.7. A *ladder* extending beyond the staff helps users to input a note which is above or below the staff. There is also a limit on the minimum size of a gesture drawn so that the gesture will not be ambiguous to the recognizer. There is no limit on its maximum size.

The number of changes in the direction of each gesture depends on the duration of the gesture's corresponding note. Thus, the shorter the duration of a note, the more changes there are in the corresponding gesture. In addition, the gestures look similar to their

equivalent notes. Although the gestures may be easy to remember, they do not consider the frequency of the use of notes in a score. The crotchet is the note that is most frequently used [anst96$_a$, anst96$_b$] and it is assigned the shortest gesture in Presto1, but that is not so in Char-rec. Thus, the speed of editing musical scores using Char-rec will not be faster than using Presto.

Char-rec is very restricted because it has gestures to input notes only; it has no gestures to delete notes or to input rests, beams, slurs, or groups of notes.

2.1.6 QuickPrint

QuickPrint is a handwriting recognizer developed by Lexicus, a division of Motorola [moto96]. QuickPrint does not have a gesture set and it steps beyond requiring users to learn gestures that the computer recognizes. Therefore, it is more successful than early handwriting recognizers that were neither accurate or popular.

QuickPrint differs from other handwriting recognizers because it assumes each user's handwriting is unique and the system has to be trained by the user to recognize the user's particular handwriting or gestures. Thus it is a writer-dependent recognizer. The training is simple and fast; it takes less than 15 minutes to teach QuickPrint to recognize a user's handwriting [whal96]. It can then translate the user's handwriting into typed text quickly and accurately [scar96]. It can even be trained to recognize Graffiti gestures in about five minutes. However, one weak point about QuickPrint is that its response time is slower than that of Graffiti [quic96].

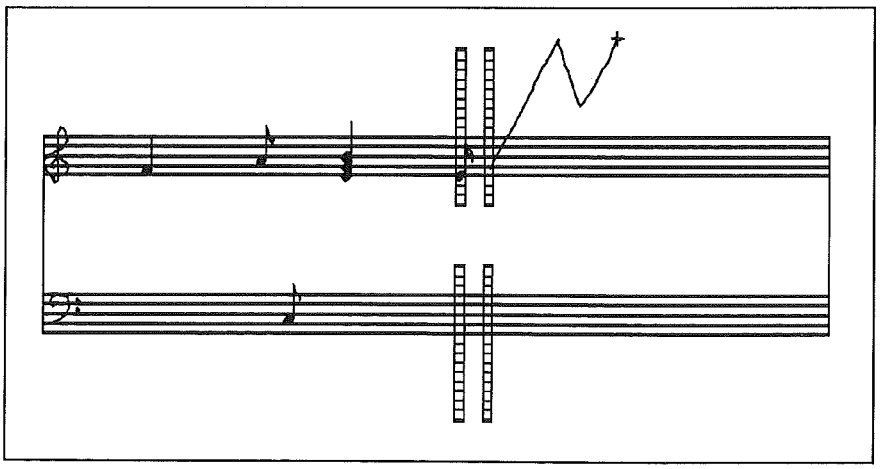


Figure 2.7: Adding a semiquaver with char-rec ([buxt79], p. 21).

2.1.7 Summary

This section described five different gesture sets and a handwriting recognizer. Unistrokes, Graffiti, and Gesture Mosaic are gesture sets that represent the Roman alphabet, and the focus of each research is on the design of the gestures. On the other hand, Pitman is a phonetic shorthand notation that already exists and it is used for effective speech transcription onto computer at the rate of dictation speed. The Pitman notation research emphasized more on the pen and the recognition of the notation.

Char-rec is a small set of gestures representing musical notes. Although the set of gestures are limited to entering notes, Char-rec is one of the few musical gesture sets available. Thus it is useful to compare it to Presto. One similarity is that both Char-rec and Presto uses the start point of a gesture to determine the position of a note that the user intends to add.

Unistrokes assigns the shortest gesture to the most common character to speed up writing, and this principle has influenced the design of Presto. Gesture Mosaic also uses very short strokes to all the letters, but it fails to make gestures easy to remember. While all the gesture sets discussed here are products from research, Graffiti and Gesture Mosaic are the only ones that are released commercially. A learning recognizer like QuickPrint would be very useful to complement an expandable Presto system as a personal music notation recognizer.

There are also other gesture sets that are not described in this thesis such as marking menus [kurt93, kurt94$_a$, tapi95] and methods of numeric entry[mcqu94, mcqu95].

2.2 Graphical music editors

Four sets of music editors that uses graphics are discussed in this section. They are the PCS terminals, Mockingbird, Finale, and Lime.

2.2.1 PCS terminals

Amando dal Molin developed a mechanical music typewriter called the Music Writer [yave87] in 1948. In the 1950s, the Music Writer was operating electrically and had a paper tape punch. In 1964, the PCS-300 Music Keypunch system was developed. In this system, dal Molin used paper tape to encode the manuscript. Each item in the manuscript is entered by selecting a *p*itch, a *c*haracter and the *s*paces (hence abbreviated

as *PCS*). The disadvantages of PCS-300 are that the electronic devices were complex and noisy, and they needed regular repairs. Thus the production of music using the system was expensive.

In 1976, dal Molin used a microprocessor and cathode ray tube technology to develop the PCS-500 Musicomp music terminal in Figure 2.8 [dalm78]. It uses graphics rather than musical semantics. This PCS is more efficient and reliable, as well as cheaper to maintain than the previous one. It has two keyboards; the left one for selecting the pitch, and the right one for entering music or text symbol. When a user enters a musical symbol, the pitch must be entered first on the pitch keyboard, and then the symbol is chosen on the other keyboard. Next the space bar is pressed the number of times required to position the symbol horizontally on the staff. Stem lengths and ledger lines are automatically generated. The beginning and end points of beams, slurs, and octava signs can be toggled on and off. There is an attached microcassette that can store 30 pages of music. The terminal described is a composer model, there is also an engraving model.

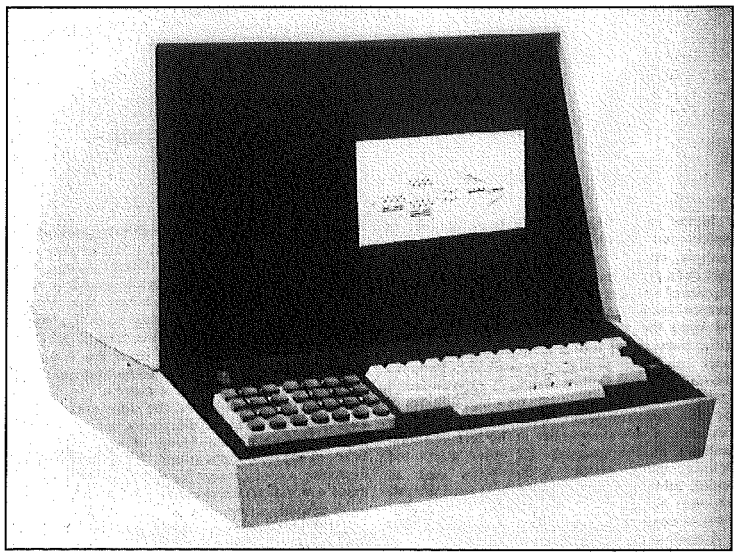


Figure 2.8: The PCS-500 Musicomp terminal ([dalm78], p. 288).

The PCS terminals are designed to be fast from the beginning. After using PCS-500 for a year, dal Molin found it fast, economical, and easy to use. However, the PCS-500 cannot print all the musical symbols that it can encode. Up to 50 percent of the symbols must be added by hand. It is also limited by the memory buffer, thus the space available to display information at any one time is small. Despite its shortcoming, dal Molin recommended that the PCS-500 would be useful to composers, arrangers, copyists, music publishers, and engravers.

2.2.2 Mockingbird

Mockingbird [maxw84] was developed in 1980 at Xerox PARC. It is a music editor for composers that combines input from a synthesizer keyboard and a mouse, as illustrated in Figure 2.9. Although it was never released commercially, it became a predecessor to many commercial interactive music editors.

Figure 2.9: The Mockingbird equipment ([maxw84], p. 385).

The design of Mockingbird was dependent on the decisions made in five important areas [yave87]. First, Mockingbird would be a composer's amanuensis rather than an automatic transcriber. Therefore, a music editor was more important than a music recognizer and the editing tools were built first. Second, Mockingbird would use a sequential data structure to handle the input and output of music. Third, Mockingbird would use what-you-see-is-what-you-get (WYSIWYG) on the user interface for display, and the mouse for input and editing. Next, Mockingbird would deal with piano music notation only, hence narrowing the scope to a manageable size. Last, Mockingbird would clearly differentiate multiple voices.

Mockingbird used a powerful computer called Xerox Dorado, shown in Figure 2.10. The only special hardware was a synthesizer keyboard used as a direct music entry and audio output tool. The mouse could be used to enter music too and was intended mainly as an editing tool. The Mockingbird program was written in Mesa, an experimental language also developed at PARC.

The screen can be scrolled with a scrollbar like those in word processors, and *thumbing* will display any part of the score requested by the user. The editor can record, edit, play, and print music. Groups of notes can be edited by selection through shading them and issuing various commands. Upon request, the program will also check whether the rhythmic time of every bar is correctly filled by the notes and rests.

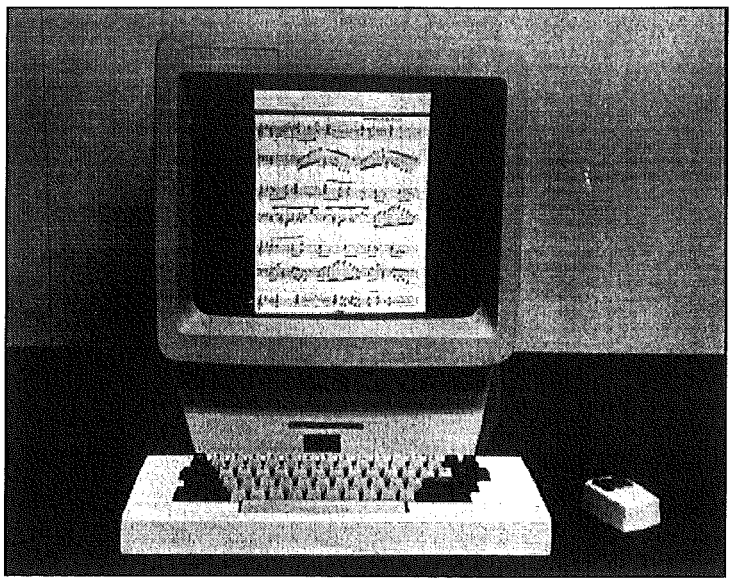


Figure 2.10: The Mockingbird computer ([maxw84], p. 387).

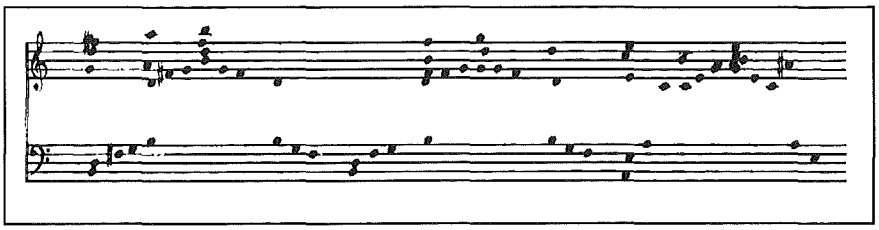


Figure 2.11: The Mockingbird piano roll input ([maxw84], p. 392).

An unusual feature is the *piano-roll* style piano keyboard entry which is viewed as a notehead-time plot shown in Figure 2.11. When the user plays notes on the synthesizer keyboard, the music captured is displayed on the plot with a staff and noteheads. Next, the user manually adds the key and time signatures, bar lines, duration of notes, and beams. When the score is complete, the user can play it back through the synthesizer and print it on a laser printer [gour86]. Some heuristics are included to help the user with the edits. These include alignment of notes that occur close to the same time; after the user has given a time signature, the program guesses the time values of the notes, groups them into chords, and assigns beams.

Mockingbird allows manual adjustments to many options, thus becoming very flexible and customizable for the user. The score output of Mockingbird is of publication quality. Moreover, Mockingbird allows the user to add a voice via synthesizer while it plays the material that is entered previously. Although the performance of playback sounds mechanical, it is clear enough for catching errors in note values and pitches. Since Mockingbird is a prototype, there are also several limitations in the system besides the limitation of editing only piano music notation, and thus it is far from complete.

2.2.3 Finale

Finale [coda89, coda90$_a$, coda90$_b$, coda93] has been accepted for several years as the most powerful notation software for the Macintosh [belk94, roth91]. However, it has also been criticised for a difficult user interface.

There are three ways to enter notation in Finale. One way is to select elements from a menu with a mouse or through the keyboard, another is through a Musical Instrument Digital Interface (MIDI) instrument, and the final way is through *HyperScribe*. HyperScribe is a real-time transcription method that converts what the user plays into musical notation, which can be edited. In addition, Finale reads MIDI files directly.

Finale has not only included all standard notation symbols, it also accepts non-standard notation by allowing users to invent their own symbols. Nearly all the symbols can be played back through MIDI. Every symbol can also be modified, cut, and pasted. When playing back on MIDI, the score is scrolled along.

Finale is a very powerful tool and its documentation is complete and comprehensive, but it is slow. Much of the time is used in redrawing the computer screen. Furthermore, the user interface is complex and the editor is cumbersome, because there are many complicated steps and procedures to complete a task. Belkin [belk94] claimed that if the window contains only one bar on one staff at a time, adjusting beam angles and reverse stems in cross-staff beaming is a "nightmare". Rothstein [roth91] found Finale extremely difficult to learn and use.

2.2.4 Lime

Lime, which is illustrated in Figure 2.12, has been under development by the Computer-based Education Research Laboratory Sound Group at the University of Illinois and Queen's University since 1974 [lime91]. It was originally built for the PLATO computer system but later it was moved to the Apple Macintosh [blos91]. Since the production of music notation cannot be completely automated because each user's needs differ, Lime aims to automate the production of music notation as much as possible but maintains flexibility at the same time [hake93].

There are two main software components in Lime: the music editor and the music formatter. A user can enter music by playing on a synthesizer instrument, typing on the keyboard, or using the mouse to select items shown on the screen. The editor analyses the user input, makes adjustments to the abstract music representation according to music notation rules, and invokes the formatter to update the screen display [blos91]. Lime can also read standard MIDI files.

Lime is developed to meet special notational needs, such as microtonal accidentals, and multiple simultaneous non-coinciding meters. It is also flexible, allowing users to customise almost all notational defaults and reapply to selected portions of the score.

There are unusual tasks that can be done in Lime too. These include stemming one chord across both staves, placing bar-lines between staves but not on them, and beaming across bar lines. Playback in Lime is complicated; some of the options available to play

one or all voices and to play softer or louder. Lime can also print a wide variety of music, including orchestral, piano and vocal scores [hake97].

Lime has its drawbacks too. It does not transcribe tuplets other than triplets, which must be entered manually. During live recording of music, it is not possible to hear the music that is already entered on other staves like in Mockingbird.

2.2.5 Summary

This section has reviewed four graphical music editors which all have WYSIWYG user interfaces. Music is entered through typing on a special keyboard on the PCS terminals, however the speed of entering music cannot be as fast as touch-typing because the user must enter at least three attributes (pitch, character and spacing) of each musical symbol. Music is entered into Mockingbird, Finale, and Lime through playing on a synthesizer instrument, using the mouse, or typing on the keyboard. Both Finale and Lime accept music through reading MIDI files too.

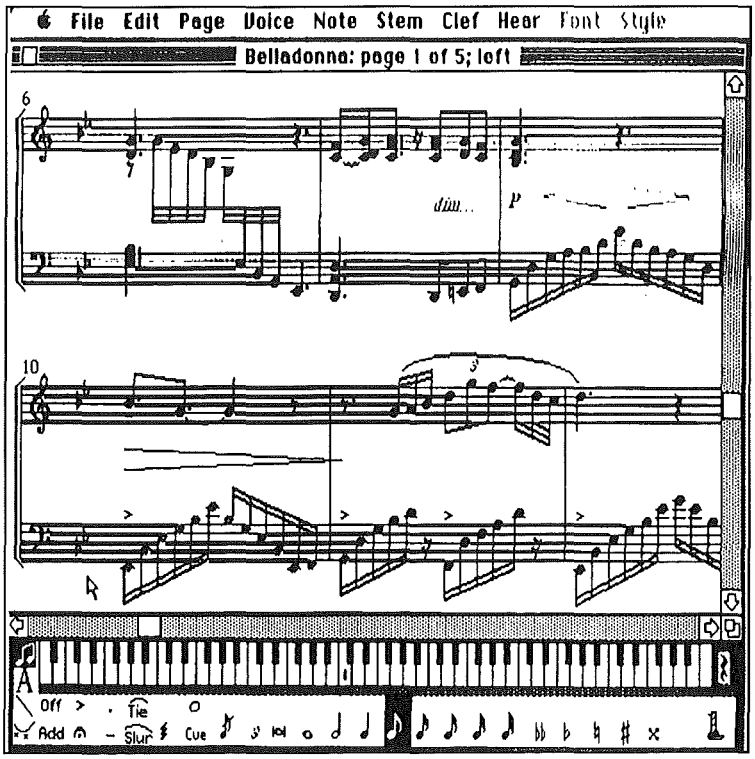


Figure 2.12: The Lime music editor ([blos94], p. 303).

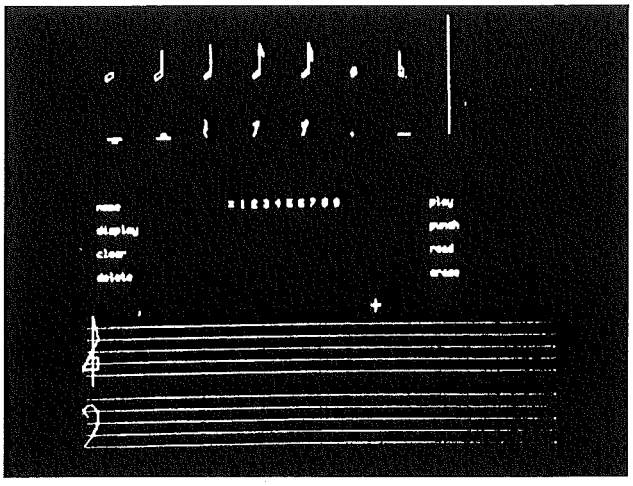


Figure 2.13: The workscope of Cantor's editor ([cant71], p. 103).

There are also many other similar editors available commercially like the Deluxe Music Construction Set [brow$_1$85], Sibelius [biel97], MusicEase [roth89$_a$, roth89$_b$], Professional Composer [gour86, yave85] and Personal Composer [mill$_2$85].

2.3 Pen-based music editors

Five sets of pen-based music editors are discussed in this section. They are Cantor's editor, SSSP editors, Gregory's Scribe, CNMAT editor, and Pitchforth's editor.

2.3.1 Cantor's editor

Cantor created one of the earliest pen-based music editors published in 1971 [cant71]. The purpose of this editor is to enter music into computer using graphics of musical notation, store the music in the computer, and edit the stored music.

Since the user is unable to scroll the score across the screen, the system has separate screens on four display scopes to display the staves on the computer. Figure 2.13 shows the screen on the first scope called the *workscope*, and Figure 2.14 shows the screen on each of the other three scopes. Pairs of staves on the screens are numbered in position from left to right and top to bottom.

A stylus held in the operator's hand in Figure 2.15 is moved over the tablet beneath the hand to enter and edit music. The operator uses the stylus to copy musical symbols in the top two rows of Figure 2.13 and places them anywhere on the staves. The computer

samples the position of the stylus and senses which symbols are to copied and where the copies were to be placed. The operator also uses the stylus to select commands in row three in Figure 2.13 to edit music. The commands include moving lines of staves among the four scopes, combining lines of staves into pieces of music, punching these pieces on paper tape, retrieving them later, displaying them on the scopes, and playing them through two audio amplifiers. It is also possible to delete one note at a time.

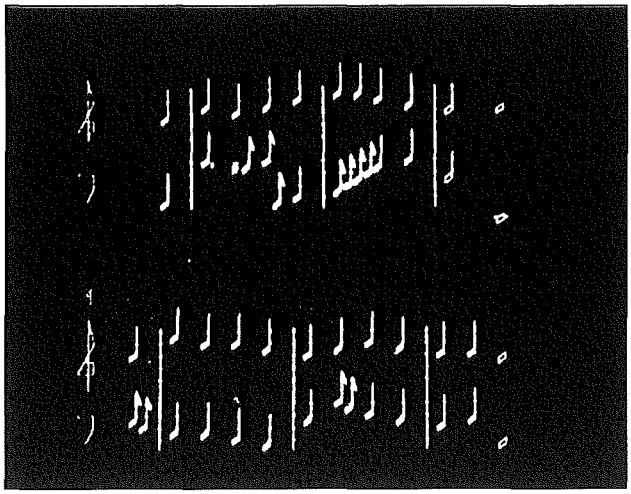


Figure 2.14: Other scopes of Cantor's editor ([cant71], p. 105).

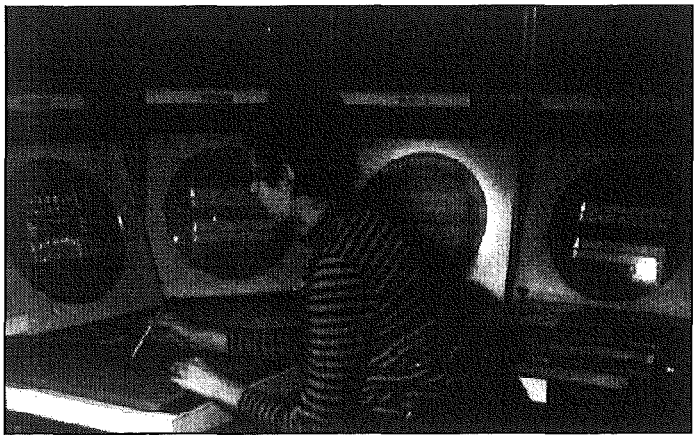


Figure 2.15: Cantor's editor in use ([cant71], p. 104).

Cantor claimed that the input of music in this display editor is faster than writing music in ink. Moreover, symbols can be erased easily and playback will help to eliminate copying errors. Nevertheless, there are limitations in this editor. Users cannot enter beams in the music, all the notes have only upward stems, accidentals must be repeated on each note, and there are limited pitches on the staves.

Cantor has proposed a few features that future editors should have. He envisaged that a music editor would have only one scope instead of four. The user could name sections of the music, clear the sections, and redisplay them. The user could also see a few bars at once, and the display could be scrolled along like a roll film in a camera. Lastly, Cantor would like to be able to create his own notations. These visions are already realized in modern systems such as Finale and Lime.

2.3.2 SSSP editors

The SSSP [buxt79] started in the 1970s at the University of Toronto. It investigates musical data structures and studies interactive computer music interfaces. Its focus is to use interactive computer graphics to help composers in the editing of musical scores. A vector-drawing graphics digitising tablet and a synthesizer are the special hardware used in this project. A few prototypes were built to aid the research and four of them are described below.

The first editor built was never used with the synthesizer. The purpose of this editor is to develop an interactive method of note input, and this method has remained as the major input tool in the project.

To input a note in the editor, the user must position the cursor over a desired pitch along a *ladder* on the staff like in Figure 2.16a. Next, the user presses and holds the cursor button in that position and a note symbol appears as a marker at that pitch, as in Figure 2.16b. At the same time, the cursor is replaced by a menu of notes with different duration. The menu moves and follows the motion of the cursor on the tablet. Hence, the roles of the cursor and the menu are reversed; instead of a stationary menu and a moving cursor, there are a moving menu and a stationary cursor. The user places the note of the desired duration over the note marker, releases the cursor button, and the note is entered on the staff, shown in Figure 2.16c.

The ledger lines, tail direction, bar-lines, and note spacing are automatically handled by the editor. The user can enter chords and polyphonic scores by positioning the cursor on

the *ladder* that already had a note. The user can also enter ties, delete notes, and make pitch corrections on the last note. There is a similar rest input tool with a menu of rests instead of notes. This editor was soon discarded after its completion.

Ludwig is the editor used together with the synthesizer, as shown in Figure 2.17. The underlying data structure to represent musical events in this editor is a single linear linked list, and the emphasis in this editor is score navigation.

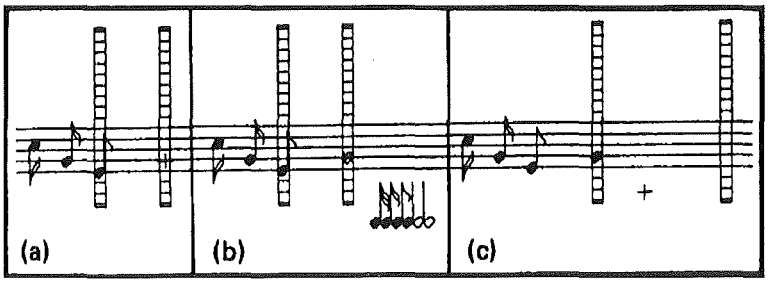


Figure 2.16: Append a note in the SSSP editor ([buxt79], p. 15).

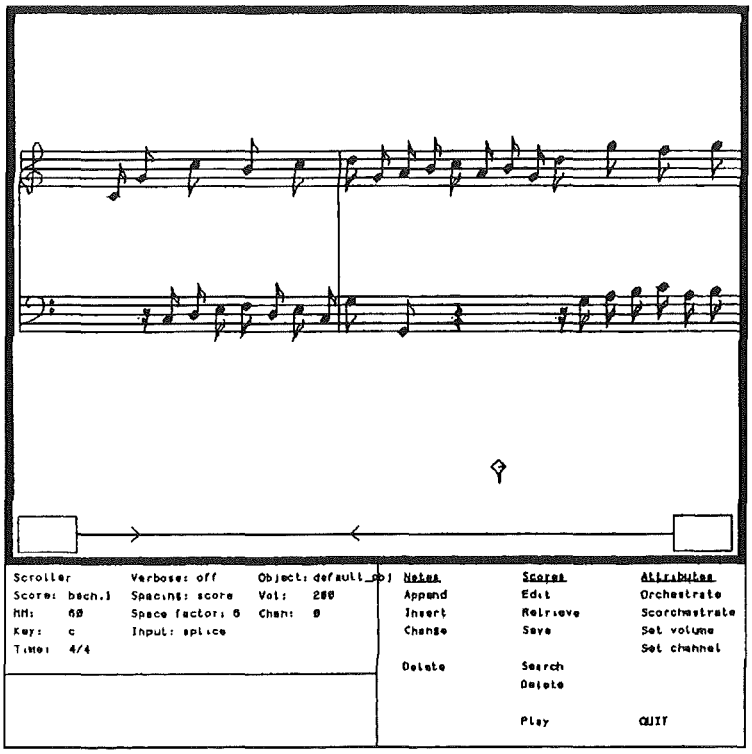


Figure 2.17: SSSP ludwig ([buxt79], p. 17).

There are two techniques to *get around* the score. One way is to scroll the score across the screen by using the hardware sliders. The second way is to select a *search* button on the screen. There is a time-line to represent the length of the score. Two brackets on the time-line indicate the portion of the score that was displayed on the screen. The user can select a portion of the score to display by positioning the cursor on the corresponding portion on the time-line and pressing the cursor button. In addition, the user can orchestrate the score by choosing the colour of an instrument out of a palette and painting it on selected notes.

The third editor was built to explore three kinds of input tools: the graphical keyboard technique, the total menu technique, and the gesture technique. The graphical keyboard technique displays a graphical piano keyboard on the screen, as illustrated in Figure 2.18. This helps the user choose a pitch and was a comfortable visual aid for a composer. There are two octaves on the keyboard, and the range is indicated by the vertical position of the *ladder* on the staff. This range can be raised or lowered by positioning the cursor and pressing the cursor button on the up or down arrow head of the cross below the staff.

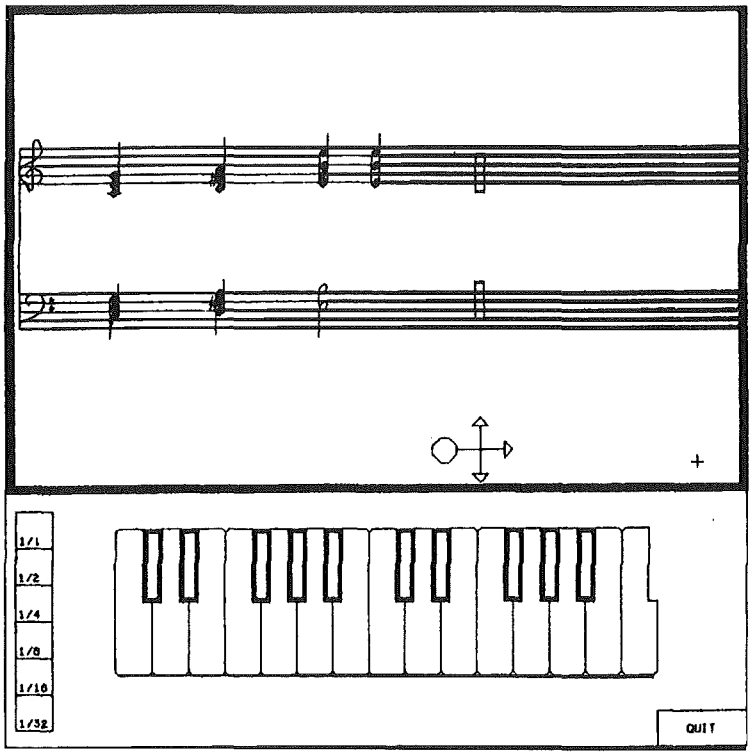


Figure 2.18: SSSP graphical keyboard technique ([buxt79], p. 19).

First, the user enters a note by positioning the cursor over the desired piano key, then by pressing and holding the cursor button. A note marker will appear on the *ladder* on the staff. Next, the user chooses the duration of this note by moving the cursor along the length of the piano key. Each piano key is divided invisibly into different segments of duration. When the cursor is on the desired duration, the button can be released and the note will be set in the staff.

The total menu technique displays menus on the screen, shown in Figure 2.19, to help the user choose the pitch and the duration of the note to input. The menus list all the options available, and present these choices to the user clearly. The third way is the gesture technique using a gesture set called *Char-rec*. This technique, which uses a character recognizer to decode note gestures into typeset notes, is presented in Section 2.1.5.

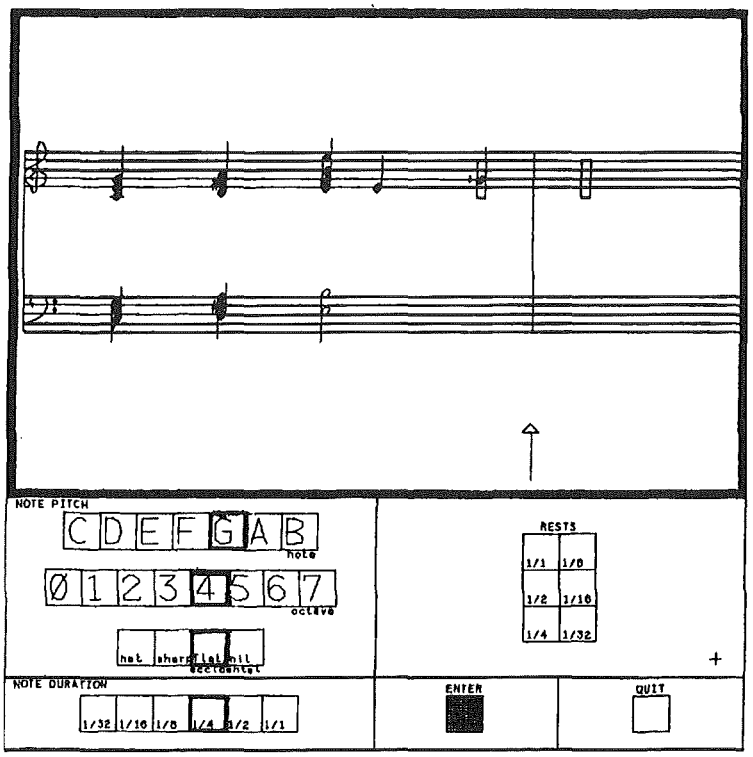


Figure 2.19: SSSP total menu technique ([buxt79], p. 20).

The graphical keyboard technique and the total menu technique are natural and conventional. The gesture technique, or Char-rec, is more fluent in the transcription of the notes than the other two techniques but is more difficult to learn. On the other hand, the total menu technique needs minimal learning, but it is also used the least in composition because many hand movements are needed to enter each note.

The fourth editor, *scriva*, integrates and expands the ideas that are developed in the previous editors. It can present a score in many ways to highlight its different aspects. In this editor, there are two methods to select notes. One way is to select them individually, and the other is to draw an enclosure around them as in Figure 2.20. To exclude some of the notes in that selection, the user can draw another enclosure around those notes. Future work in the SSSP includes developing editors that handle tree-structured scores and that scroll the scores during playback. The SSSP also wants to test the editors with larger scores.

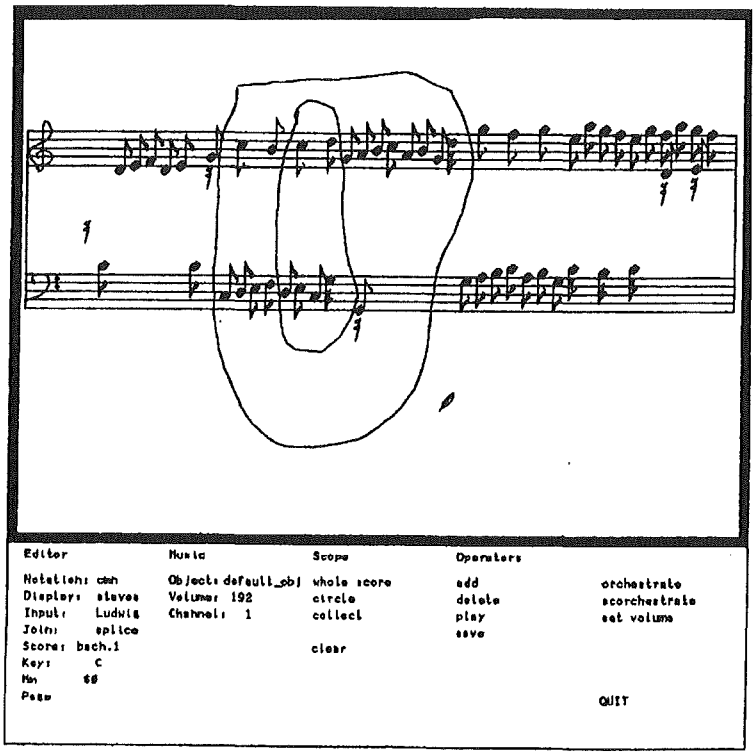


Figure 2.20: Selection in SSSP scriva ([buxt79], p. 24).

2.3.3 Gregory's Scribe

Gregory's Scribe [craw83] published in 1983 is an inexpensive graphics program for laying out pre-1600 music notation. Figure 2.21 shows a sample of pre-1600 music produced by Gregory's Scribe. It utilises a graphics tablet and a pen to select commands similar to those of a drawing package. The symbols that are available in Gregory's Scribe are shown in Figure 2.22. The system is limited to producing music notation from before 1600, because this notation has only straight lines which are suitable for the dot-matrix graphics used.

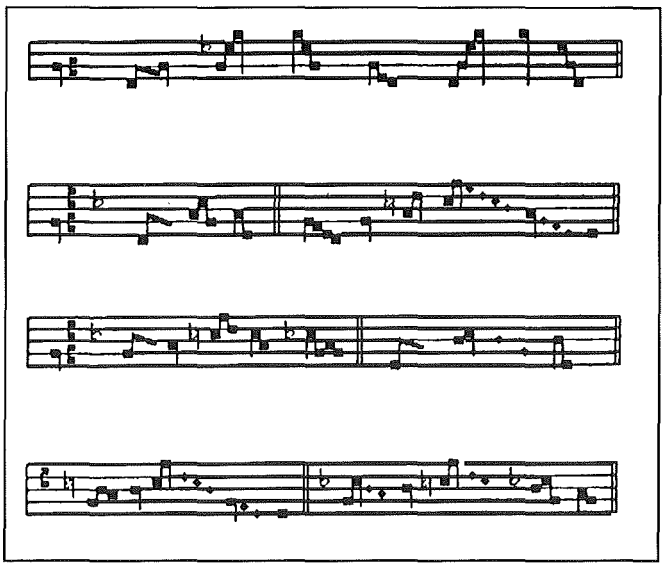


Figure 2.21: Score produced by Gregory's Scribe ([craw83], p. 24).

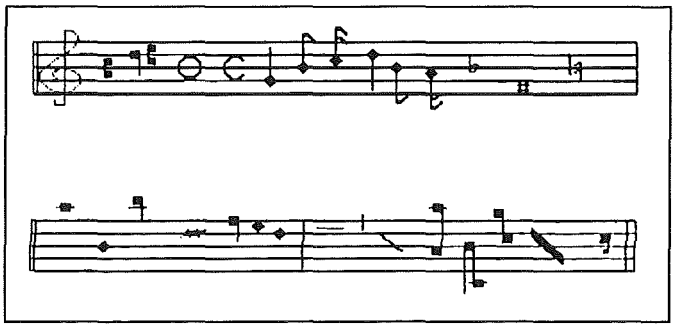


Figure 2.22: Gregory's Scribe characters ([craw83] , p. 23).

File controls in the program are given at the keyboard, but most commands are in graphics mode and are entered with the graphics pen and tablet. To input a symbol, the user chooses a symbol in Figure 2.22 and presses the desired position on the staff. The chosen symbol will then be copied there. The user can press *b* or *w* on the keyboard to create solid black or white note shapes, or press the space bar to move the next symbol ahead. If the user does not press the space bar, the program will draw the next symbol in the position where the pen is pressed on the tablet. Users of Gregory's Scribe at the University of Michigan were experienced music copyists and they agreed that using Gregory's Scribe is faster and neater than using the ink pen.

2.3.4 CNMAT system

The Center for New Music and Audio Technologies (CNMAT) designed a music notation system [lero94] that uses a pen and handwritten symbols for fast and flexible input. Like Presto, the objective of the CNMAT system is to rapidly input musical ideas and store them so that they can be processed further. Such processing includes printing, using with a PDA, and using for synthesis control and performance. The designers of the system claimed that the pen provides more precise gestures than the mouse due to the direct feedback produced from a touch-sensitive display. This system is designed for composers, but not engravers.

There are six user requirements in the design of the system:

- To make entering and editing music notation simple and fast. In other words, it is analogous to composers and copyists using shorthand music notation, but obtaining engraver-quality results.
- To be able to edit music at both local and global levels.
- To support the performance of the system's music through computer and other electronic instruments, support a variety of music notations, and support different user interfaces.
- To fully document the process of composing so that the user can have access to all versions of the composition.
- To correspond the design of the system to paper and pencil. Heuristics in the program should not interfere with user input.
- To have a robust environment in the system.

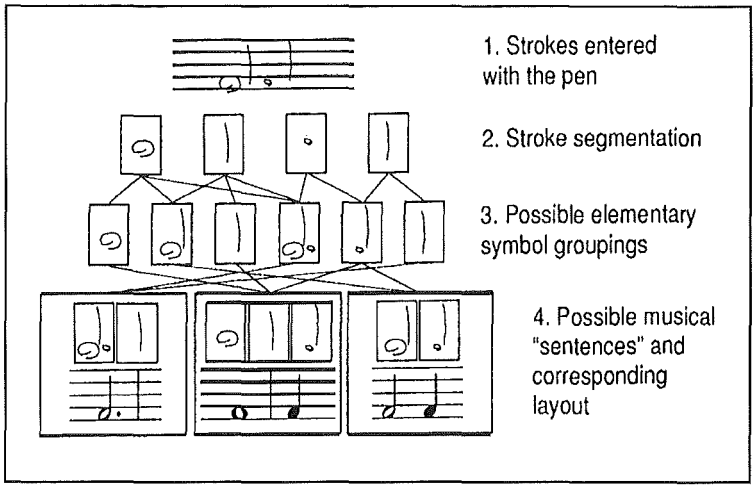


Figure 2.23: Recognition process in CNMAT editor ([lero94], fig. 3).

The history mechanism in the system not only documents the final state of the score, but it also records the entire process including the stroke input. Therefore, the user can retrieve any previous state of the composition and create a new score from there. The score is saved as a snapshot of the composition state at that time. Although there is no undo operation to restore a previous state, the history mechanism is used to traverse the sequence of states from the original up to the current state.

There are two independent parts in the system: recognition and representation. These parts are designed using the object oriented Model-View-Controller paradigm. The system uses the bottom-up approach to recognize gestures. After the user draws a few strokes in Step 1 in Figure 2.23, the user presses a button to start the recognition. First, the system segments and labels the strokes entered by the pen in Step 2. Next, it recursively groups the strokes into possible combinations in Step 3, and presents the most probable one to the user in Step 4. If it is accepted, the combination is displayed on the staff. There are three editing modes in the system; the user can input a stroke directly, edit that stroke before recognition, or edit symbols that are already recognized.

2.3.5 Pitchforth's system

Pitchforth [pitc94] designed and implemented a technique [pitc94] to train a music editor to recognize the user's personal handwritten music notation in real time. The scope of the system is to deal with music input, and particular parts of some existing recognition methods are integrated into the recognizer. Pitchforth trained and tested the

system with standard musical notation, where testing samples are shown in Figure 2.24, but the system can also be trained to recognize shorthand equivalents. Pitchforth intended the system to complement a package which facilitates the printing and playing of music.

When the user draws a gesture and leaves the tablet untouched for a sixth of a second, the recognizer assumes that the gesture is finished. Then, the recognizer draws a bounding box around the gesture, processes and compares it to the library. If the gesture is known, the program states so and shows the number of matches found. If it is unknown, the name of the gesture can be entered from the keyboard and the gesture is stored. The method of decomposing gestures into a series of vectors is used to recognize gestures. The system can learn new gestures with approximately eight samples. The results are presented to the user in text.

There are several problems in the system. Pitch recognition of notes is harder and the design of this part of the system is not complete at the time when the project ended. The system cannot capture the stem direction of notes, and it must estimate on which end of the stem the head is.

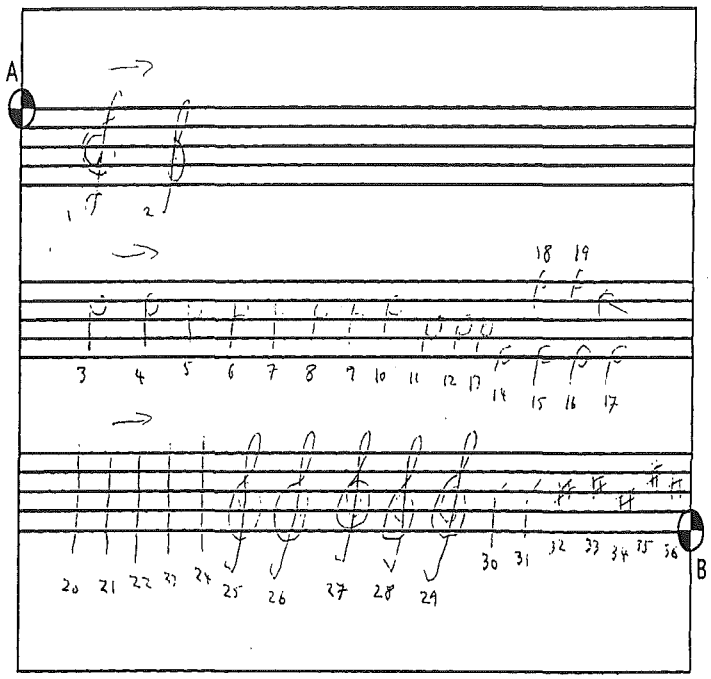


Figure 2.24: Testing samples in Pitchforth's editor ([pitc94], p. 11).

2.3.6 Summary

This section has discussed five pen-based music editors. Among these editors, Cantor's editor is the earliest one, but it has many technological constraints such as screens that cannot be scrolled and a stylus that only copies and pastes symbols. Some of these issues are addressed in the SSSP editors. They include scrolling the screen and providing a time-line of the score to allow the user to go to a specific location. The SSSP has also explored three ways of entering notes; by selecting on a music keyboard on the screen, choosing from menus, and drawing gestures. One of the SSSP editors allows selection by drawing a circle around the desired notes. Gregory's Scribe is an editor designed for entering pre-1600 music notation. Like Cantor's editor, this editor uses a stylus to copy and paste symbols onto the staff.

The CNMAT system is closest to Presto among the other pen-based music editors, mainly because both systems have the objective of using pen entry as the fastest way of inputting music into the computer. However, the CNMAT system uses complete music notation and not shorthand gestures for pen entry. Presto can be expanded into a learning music editor by integrating Pitchforth's system.

2.4 Summary

There are many features from the gesture sets and the editors discussed in this chapter that Presto can incorporate. In Unistrokes, all the gestures are relatively simple and the frequency of the use of each letter determines which gesture is to be assigned to that letter. If a letter is used often, that letter is assigned with a short gesture. On the other hand, if a letter is seldom used, that letter is assigned with a longer gesture. The design of Presto already uses this principle, and this principle should be followed in the development of other new gestures in Presto.

Unistrokes, Graffiti, and Gesture Mosaic all prove that users can learn at least 26 new gestures, although some gestures are inverted, rotated and reused. This is an encouragement that Presto can expand its present number of gestures to represent other less common musical symbols and can still remain easy to remember and fast to draw.

Both Gesture Mosaic and Char-rec use a template to guide the user to draw the gesture. Presto can use a template that is similar to the *mosaic* in Gesture Mosaic to guide the

drawing of the strokes in the gestures. Presto can also use a scaffolding like the *ladder* in Char-rec to help users locate the accurate location of notes that are not on the staff.

In Gesture Mosaic, writing letters has an audio feedback different from that when writing numbers. Similarly, Presto can produce different sounds when different musical symbols are drawn. For example, the sound for drawing a note will be different to the sound for drawing a rest. Apart from the sound of each symbol, Presto can also let the user hear a playback of the score like that of Mockingbird.

Mockingbird has a command to go to any part of the score a user wants to, and SSSP *ludwig* also has that command and adds a time-line to show the part of the score that the user is in. It will be helpful for the user to be able to activate a *go to* command and look at a time-line in Presto.

If Presto has a history mechanism like that in the CNMAT system, it can let the user backtrack the steps of entering and editing the score to the beginning state of the system. Presto can also be complemented by trainable recognizers, such as QuickPrint and Pitchforth's system to recognize the user's personal gestures accurately.

Chapter 3

Gestures

To make the pen-input music system easier to use, pen gestures must be fast, short, easy to draw, and mnemonic or easy to remember. The details of these characteristics are listed in Section 1.5.

This chapter first examines some issues in gesture recognition in Presto1, the gesture set that Anstice [anst96$_a$, anst96$_b$] designed. A gesture is a combination of electronically-produced strokes which executes commands, a gesture set is a group of gestures designed for use in a program. The chapter also looks at some of the gestures in Presto1, then evaluates, redesigns, and redefines them mnemonically in Presto2, the gesture set which the author developed from Presto1. All the gestures in Presto2 are presented in Appendix A. Buxton et al. [baec95$_a$] discusses gestures and markings in detail. The gestures discussed in this chapter include inserting rests and notes into the score, changing the stem directions of notes, changing the duration of notes and rests, and inserting double bar-lines. These gestures are implemented in MusEd2, the system that is developed to test Presto2.

3.1 Issues in gesture recognition

A pen gesture in a pen-based editor can be recognized by:

- its shape, for example a straight line or a curve;
- its direction, for example drawing to the right or to the left;
- its context, for example whether it is drawn on a notehead, a rest or in blank space;

- its speed, that is fast or slow;
- its length, that is long or short; and
- whether a button exists on the barrel of the pen and is being pressed.

The recognizer of MusEd1, the system that was previously designed to test Presto1, recognizes gestures in eight directions as shown in Figure 3.1 [anst96$_a$]. Each direction is defined by a segment with a range of angles for recognizing that direction. Segments one, three, five, and seven subtend 52 degrees each, and segments two, four, six, and eight subtend 38 degrees each. The odd-numbered segments have larger angles than the even-numbered segments because gestures in the former segments are more common. These eight directions allow gestures to consist of horizontal, vertical, and diagonal lines, but diagonal lines posed three problems in MusEd1.

First, 60 percent of the subjects in the beam survey, which is discussed in Section 5.4, had difficulty drawing the minim gesture. This gesture, as shown in Table 3.1, needs a line in segment four then a line in segment six in Figure 3.1. MusEd1 either misrecognized or did not recognize the gesture, because the system allowed only a small tolerance for the lines of that gesture and would not accept a sloppy gesture. Moreover, two diagonal lines mean that the user must draw an accepted diagonal line twice. This is a similar problem for the gestures that change the duration of a note or a rest, and the gestures that change the stem of a note upwards or downwards.

Second, the gestures for changing the stem direction of a note and those for changing the duration of a note or a rest are similar in shape, as shown in Table 3.1, and context, that is they start on the notehead. The gesture for changing the stem upwards consists of a line in segment two followed by a line in segment eight, while the gesture for halving the duration consists of a line in segment six or seven and a line in segment four or three in Figure 3.1. The gesture for changing the stem downwards has a line in segment four then a line in segment six, and the gesture that doubles the duration has a line in segment three or four and a line in segment six or seven in Figure 3.1. The differences between the two sets of gestures are in the angle and the directions. The angle between the two diagonal lines of the stem direction gestures are larger than that of the duration gestures. This may cause the user to get confused between the two sets of gestures.

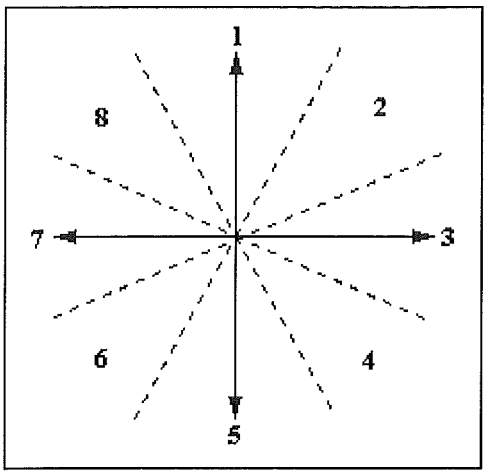


Figure 3.1: Eight directions of recognition in Presto1.

Symbol / effect	**Gesture**	**Context**
Rests	↓	Draw on the staff, and select from the pop-up menu.
crotchet	●	Dot on the pitch.
crotchet with upward stem	↑	Start on the pitch, and draw a middle line.
crotchet with downward stem	↓	Start on the pitch, and draw a middle line.
minim	↘	Start on the pitch.
Set stem upward	↗	Start on the notehead.
Set stem downward	↘	Start on the notehead.
Double duration	↘	Start on the notehead or rest.
Halve duration	↙	Start on the notehead or rest.

Table 3.1: Some current gestures in the Presto1.

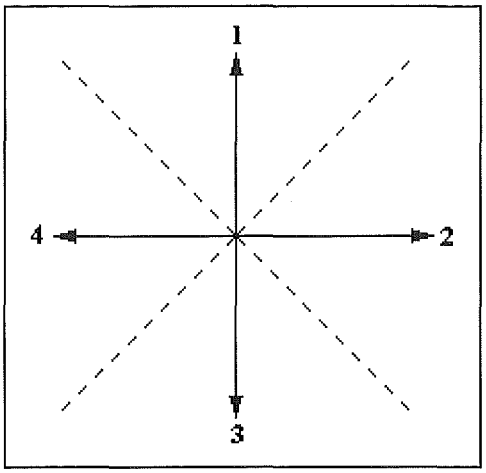


Figure 3.2: Four directions of recognition in Presto2.

The third problem exists because the downward stem gesture is drawn in the same direction as the double gesture, only with a difference in the angle. Both gestures consist of one line in segment four and another line in segment six in Figure 3.1. Thus, it is difficult for the user to draw the gestures sloppily, because the system might misrecognize one gesture for another.

The eight directions of recognition for Presto1 can be simplified into four directions as shown in Figure 3.2, such that the angle of each segment will be 90 degrees, which is larger than the segments in eight directions. This means that gestures can consist of only horizontal and vertical lines, but not diagonal lines. Gestures using these four directions can also be more sloppy than those using eight directions. However, gestures in Presto1 that consist of diagonal lines will have to be redesigned and the number of gestures that are available will decrease.

Although the way of recognizing gestures in only four directions seems less powerful than that of recognizing in eight directions, the four-direction recognizer might perform better. For example, if a user intends to draw a horizontal right flick but draws it with a slight slope, the eight-direction recognizer will categorize the flick as a gesture in segment two in Figure 3.1. In Presto1, a flick in segment one adds a sharp to a note and a flick in segment three adds a durational dot to a note [anst96$_a$]. However, a flick in segment two is not associated with any action because the system does not know which command the user intends, thus MusEd1 will treat that gesture as an error. When the same flick is drawn in MusEd2, the four-direction recognizer will classify the flick as a

gesture in segment one if the slope of the flick is steeper than 45 degrees in Figure 3.2, and as a gesture in segment two if the slope is flatter than 45 degrees. In Presto2, a flick in segment one adds a sharp to a note and a flick in segment two adds a dot to a note. The gestures in Presto2 are listed in Appendix A.

If a gesture is delimited to straight lines and two changes in direction, there will be 17 shapes available. Figure 3.3 shows the 17 shapes. The shapes are drawn in the direction of the arrowhead and a number beside a line is the number of the segment where that line is. An arrowhead in the middle of a stroke indicates that the shape is drawn first in the reverse direction then in the direction of the arrowhead. Sixteen of these shapes, excluding the dot, can be drawn with long or short length and drawn fast or slow. All 17 shapes can also be drawn with the button on the pen barrel pressed or not and drawn with different context. Overall, more than 260 gestures can be produced from the four-direction recognizer.

The gestures in Presto1 do not use speed, but use length and context to be unique. Speed and length often depend on the user and the system. A stroke that is fast to one user may not seem to be fast to another user or the system. This is similar to slow, short or long strokes. The length of a stroke can also be affected by the size of the staff, which is discussed in Section 4.5. Thus, recognition of gestures that uses speed and length may not match the user's expectations. On the other hand, recognition of gestures in different context is more reliable than using speed or length. The system knows where the symbols are and the user can see the symbols clearly, although the system may need some tolerance when the gestures are drawn. The button on the pen barrel is already assigned exclusively for the delete gesture, which is discussed in Section 4.1.

For these reasons, Presto2 will mainly use context for creating different gestures, but not speed and length. Any gestures in Presto1 that are also used in Presto2 will not have a required length; these gestures can be short, medium, or long. These gestures are those that change the pitch of a note, add or remove a durational dot, and draw a bar-line. Furthermore, the corner between two lines in a stroke need not be sharp; a sloppy gesture with a rounded corner can also be recognized accurately.

3.2 Notes

In this section, gestures for drawing a crotchet with a fixed stem direction and a minim are examined.

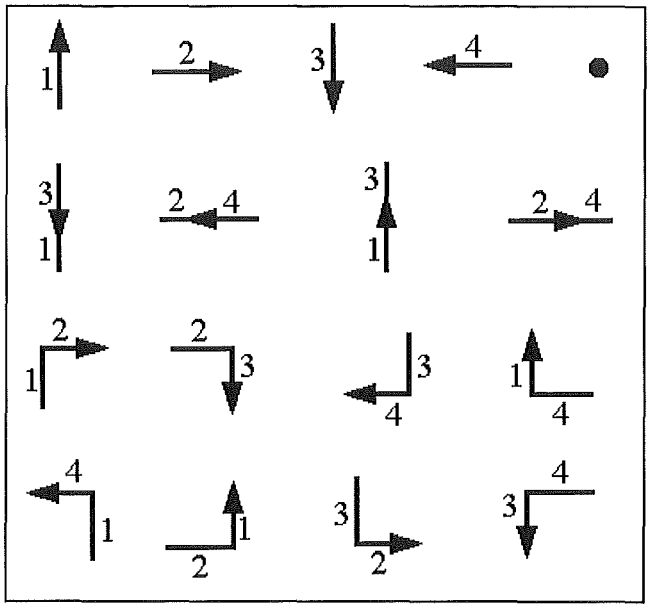


Figure 3.3: Seventeen gestures from the four directions.

3.2.1 Design

Sometimes, the user may need to draw a crotchet with a stem that does not go in the default direction. In Presto1, the user has to draw a line with middle length to insert such a crotchet, as shown in Table 3.1. This is difficult to draw because it has a similar length to the bar-line gesture. Moreover, Presto2 does not use length to differentiate gestures, thus it would be more appropriate to treat any length of line drawn vertically as the bar-line gesture. As a result, this gesture for drawing a crotchet with fixed stem direction needs to be redesigned.

The gesture in Presto1 can be modified into a new shape by not restricting the length of the stroke and by repeating that line in a different direction. For example, drawing a line from bottom to top then back to bottom can insert a crotchet with an upward stem. This is like drawing an upward stem from the head of the note. A line can be drawn from top to bottom then back to top to insert a crotchet with a downward stem; a gesture similar to drawing a downward stem from the head of the note. In Table 3.2, the symbol of the gesture for drawing a crotchet with a fixed stem direction is a vertical line with an arrowhead in the middle. The optional symbols in the table mean that the two lines of the gesture need not be exactly parallel and can have an angle smaller than 90 degrees,

allowing sloppy gestures to be recognized. The two lines need not be the same length. The start of the gesture will indicate the pitch of the note.

In Presto1, the direct way to insert a minim is to draw two diagonal lines with an acute angle between them, as shown in Table 3.1. However, gestures cannot consist of diagonal lines as described in Section 3.1, thus this minim gesture cannot be used.

The difference between the crotchet and the minim is in the head of the note; the crotchet has a black head while the minim has a white or hollow head. Since a dot on the staff on the desired pitch inserts a crotchet, a dot on the black head of a crotchet can turn the head white and change the crotchet into a minim, as shown in Table 3.2. The user can also view a minim as drawing a double dot directly on the staff. The dot gesture on the head of a note is further developed in Section 3.4.2.

Breves and semibreves are less common musical symbols in music [anst96$_a$], thus they can be obtained indirectly by increasing the duration of a minim. The user can also get quavers and notes of shorter duration by decreasing the duration of a crotchet. The gestures for changing the duration of a note or a rest are discussed in Section 3.4.2.

In some pieces of music, the most common musical symbol is not the crotchet, which is assumed to be most common in Anstice [anst96$_a$]. Thus, MusEd2 can include a menu that has an option to use the shortest gesture, the dot gesture, to insert a particular musical symbol other than the crotchet.

Symbol / effect	**Gesture**	**Context**
crotchet with upward stem	or or	Start on the pitch.
crotchet with downward stem	or or	Start on the pitch.
minim		Dot on the crotchet notehead.

Table 3.2: Proposed note gestures in the Presto2.

3.2.2 Implementation

The gestures that insert a crotchet with a fixed stem direction are implemented into MusEd2. The gesture that adds a crotchet with an upward stem consists of a line in segment one followed by another line in segment three in Figure 3.2. The gesture that adds a crotchet with a downward stem has a line in segment three with another line in segment one. The dot gesture is already recognized in MusEd1, thus if the dot gesture is drawn on the head of a crotchet in MusEd2, the crotchet will change into a minim. Users will also get a minim if they draw two dots in quick succession. The details of the gestures are in Appendix B.

An option that allows the user to choose a particular note to be inserted when drawing the dot gesture is implemented into the main menu of MusEd2, as shown in Figure 3.4. The crotchet is set as the default note. If the default note changes, for example to the minim, the dot and the fixed stem gestures will insert the note chosen, which is a minim in this case. The double dot gesture will insert a note with double the duration of the chosen note. Rests are not included in the option menu because the dot and the stem direction gestures are the only gestures that insert notes. If a rest is chosen as the default musical symbol, the user will not be able to insert a note at all.

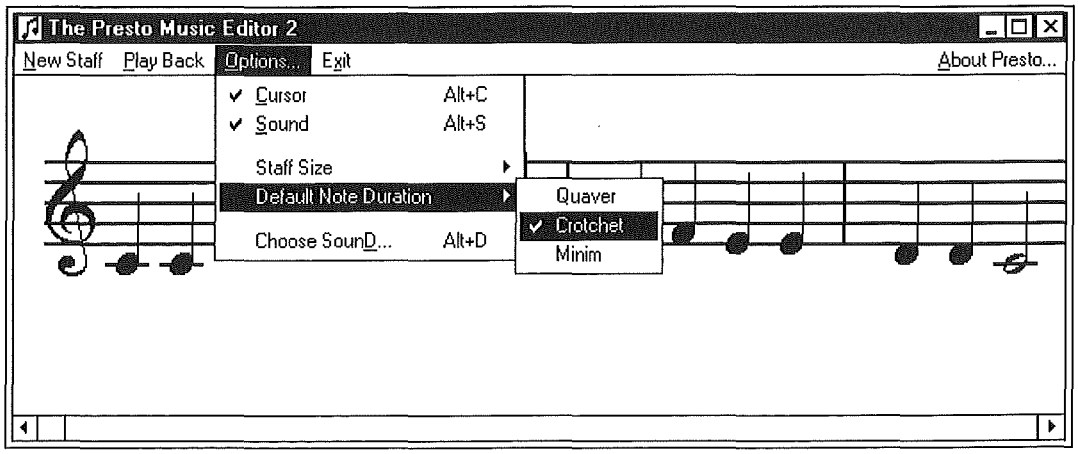


Figure 3.4: Option to choose note duration in MusEd2.

Symbol / effect	Mnemonic	Gesture	Context
quaver rest			Draw on the staff.
semiquaver rest			Draw a short length over the leg of the rest with longer duration.
demisemiquaver rest			
hemidemisemiquaver rest			
crotchet rest			Draw on the staff.
minim rest			Draw on the staff.
semibreve rest			Draw on the staff.

Table 3.3 : Proposed rest gestures in the Presto2.

3.3 Rests

This section examines the gestures for drawing rests, from a semibreve rest to a hemidemisemiquaver rest.

3.3.1 Design

In Presto1, rests are inserted into the score by selecting the appropriate rest from a pop-up menu [anst96$_a$]. This menu is activated by the gesture that looks like a right-angled *7* and drawn from left to right then down, as shown in Table 3.1. This way of inserting rests is slower than drawing a gesture directly to insert different rests, but it is only intended as a temporary gesture. Therefore, more direct gestures for rests are needed.

The gesture that activates the rest menu also looks like the quaver rest and can be used to insert the quaver rest directly, as shown in Table 3.3, instead of choosing it from a menu. In this way, the gesture for adding a quaver rest is more mnemonic and faster than using a pop-up menu.

Presto1 allows the user to change a quaver into a semiquaver if the user draws a line over the stem of the quaver [anst96$_a$]. Similarly, the user can also change a quaver rest into a semiquaver rest by drawing a line over the *leg* of the quaver rest. Since there is no limit in length, the user can change a group of quaver rests in one gesture by drawing

over the legs of those rests. Graphically, this gesture adds a flag to the quaver rest, as shown in Table 3.3. The user can also draw lines to a semiquaver rest and demisemiquaver rest to change them into a demisemiquaver rest and hemidemisemiquaver rest respectively. The line gesture drawn over symbols is developed in Section 3.4.2.

The crotchet rest should have its own gesture too. The crotchet rest is sometimes written as 𝄽 instead of 𝄾, but it is no longer standard in common music notation. However, a gesture similar to it could be used to insert the crotchet rest, as shown in Table 3.3. In other words, this gesture is a right-angled but reversed *7*, and it is drawn from right to left then down. This gesture is also the reversed version of the quaver gesture.

When musicians draw a minim rest or a semibreve rest on paper, they do not usually draw the filled rectangle. Instead, they draw a short and thick line either sitting on or hanging from a staff line. According to the way the minim rest or the semibreve rest is drawn, the gestures for adding these rests can be a stroke to the right or a stroke to the left. However, these gestures are already used for adding and removing a dot to a note or a rest, redoing and undoing a command which are discussed in Section 4.2, drawing a beam discussed in Chapter 5, and halving the duration of a note or a rest discussed in Section 3.4.2. These gestures are also listed in Appendix A.

A more appropriate gesture for drawing a minim rest is to use the right-angled *7*, flipping and rotating it into a mnemonic shape. In this way, the gestures for drawing rests are in a similar shape. The gesture for inserting a minim rest can consist of two lines; the first line is drawn from bottom to top representing the left edge of the minim rest, and the second line is drawn from left to right representing the top edge, as shown in Table 3.3. This suggests that the semibreve rest can also be drawn by two lines, going from top to bottom then from left to right, representing the left and the bottom edges of the semibreve rest, as shown in Table 3.3. The semibreve rest can be changed into a breve rest, a rare symbol in scores [anst96$_a$], by drawing the gesture that doubles the duration of a rest, which is discussed in Section 3.4.2.

3.3.2 Implementation

The rest gestures are implemented into MusEd2. The quaver rest gesture has a line in segment two and a line in segment three in Figure 3.2. The crotchet rest gesture consists of a line in segment four followed by a line in segment three. The minim rest gesture has

a line in segment one then a line in segment two. The semibreve rest gesture requires a line in segment three and a line in segment two.

Users can use the horizontal line gesture to the right or to the left to change a quaver to a semiquaver in MusEd1. Similarly, MusEd2 lets users change a quaver rest to a semiquaver rest with the horizontal line gesture. Drawing a horizontal line over a group of rests in MusEd2 is similar to drawing a beam over a group of notes in MusEd1, which is discussed in Chapter 5, but the rests are halved and do not have a beam on them. The same tolerances of the beam gesture on notes, which is described in Section 5.7, are used on rests in MusEd2. The details of the recognition of these gestures are in Appendix B.

3.4 Other gestures

In this section, three sets of gestures are redesigned and redefined. They are the gestures for changing the stem direction of a note, the gestures for doubling and halving notes or rests, and the gesture for inserting a double bar-line.

3.4.1 Stem direction of notes

The user may want to change the stem direction of notes. The gestures available for setting the stem upwards or downwards in Presto1 consists of two diagonal lines with an obtuse angle between them, as shown in Table 3.1. However, as discussed in Section 3.1, Presto2 does not recognize diagonal lines including the stem direction gestures.

Since changing the stem direction of a note is similar to inserting a crotchet with a fixed stem direction, the same gesture can be used to change the direction of the stem if it is drawn on the notehead. A gesture with up then down lines starting on a notehead with a downward stem can change the stem upwards, and gesture with down then up lines on a notehead with an upward stem can change the stem downwards, as shown in Table 3.4.

3.4.2 Duration of notes and rests

The user can change the duration of a note or a rest in Presto by factors of two. For example, doubling a minim will change it into a semibreve, and halving a minim will change it into a crotchet. The gestures for changing the duration of a note or a rest in Presto1 uses diagonal lines with an acute angle between them, as shown in Table 3.1.

However, Presto2 will not recognize diagonal lines in the double and the halve gestures, as discussed in Section 3.1.

The solution for Presto2 is to close up the angle between the two diagonal lines, causing these two lines to become straight lines but in different directions. This new gesture is shown in Table 3.4. In this way, the two lines are a repetition of each other but they need not be exactly parallel; they can be slightly slanted and allow an angle smaller than 90 degrees between them. The user can remember these two gestures by thinking that drawing to the right first and then back would mean increasing the duration of the symbol, and drawing to the left first and back would be decreasing the duration of the symbol. The gestures must start on a notehead or a rest to indicate which symbol to double or halve.

We can also use the dot gesture to double the duration of a note or a rest if it is drawn on a notehead or a rest, since a dot on the head of a crotchet changes it into a minim, as discussed in Section 3.2. Similarly, the line gesture drawn over a note or a rest can halve the duration of the symbol. These gestures are shown in Table 3.4.

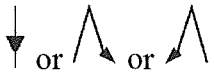
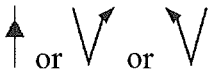
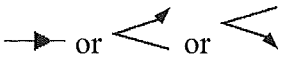
Symbol / effect	**Gesture**	**Example**	**Context**
Set stem upward			Start on the notehead.
Set stem downward			Start on the notehead.
Double duration			Start on the notehead or rest.
			Dot on the notehead or rest.
Halve duration			Start on the notehead or rest.
			Draw over the note or rest.
double bar-line			Draw next to a single bar-line.

Table 3.4: Other proposed gestures in the Presto2.

3.4.3 Double bar-line

A double bar-line indicates the end of a piece of music, the end of a major section, repeats when two or four dots are also added, or that the key signature is changed [tayl94]. Thus, at least one double bar-line is needed in each score. The double bar-line is not available in Presto1. When subjects in the beam survey discussed in Section 5.4 were experimenting with MusEd1, three out of ten subjects asked for a double bar-line gesture. Thus, although the number of double bar-lines is not as many as the number of other symbols [anst96$_a$], double bar-lines are still essential.

Since the double bar-line is an extension of the single bar-line, the gesture for inserting the double bar-line in Presto2 can be similar to that of the single bar-line. When the user draws a line near a single bar-line in MusEd1, the editor will insert an empty bar in the score. In MusEd2, this gesture can be used to change the single bar-line into a double bar-line if it is drawn on or near the single bar-line, as shown in Table 3.4. Therefore, the user can insert a double bar-line by drawing two vertical lines consecutively. MusEd2 will insert an empty bar if another line is drawn on or near the double bar-line.

There should be a *tolerance* for the double bar-line gesture; a line drawn outside this tolerance will insert an empty bar. In Figure 3.5, the area between the two lines is the tolerance for the double bar-line. The notes in the figure represents musical symbols that can be placed next to a bar-line. If p is the proportion of the space between the note and the single bar-line that the line gesture will change the single bar-line into a double bar-line, then $1\text{-}p$ is the proportion of that space that the gesture will insert an empty bar. The value of p can be zero where the double bar-line cannot be drawn, one where the double bar-line can be inserted anywhere between the single bar-line and another symbol, or between zero and one like in Figure 3.5.

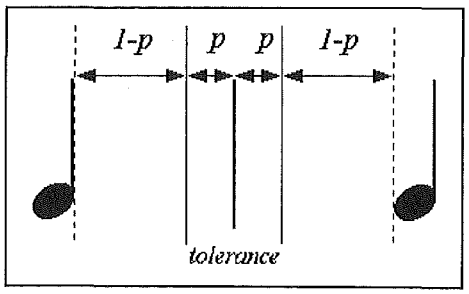


Figure 3.5: *Tolerance* of the double bar-line gesture.

3.4.4 Implementation

The upward stem and the downward stem gestures are the same as the crotchet with upward stem and crotchet with downward stem gestures except that the former gestures change the stem of an existing note when their start point is drawn on the notehead.

Both sets of double and halve gestures are implemented in MusEd2. The limits of the double and halve gestures are subject to the range of notes and rests used in Presto2, that is from breve to hemidemisemiquaver and similar duration for their equivalent rests. One of the double gesture is a line in segment two and a line in segment four of Figure 3.2, and the halve gesture is a line in segment four then a line in segment two. The implementation of the alternative double and halve gestures, which are a dot and a line, are discussed in Section 3.2.2 and Section 3.3.2 respectively. These gestures can be drawn on all the notes and rests available in MusEd2.

Drawing the double bar-line gesture in MusEd2 is the same as drawing the single bar-line gesture in MusEd1, but the user must draw it on or near a single bar-line. Since the user may need to draw a double bar-line or insert an empty bar in MusEd2, the value of p in Figure 3.5 is one-third; this allows for considerable tolerance. Appendix B contains the details of the recognition of these gestures.

3.5 Summary

This chapter has reported work that changed the way Presto recognizes gestures; the eight directions in the recognizer is reduced to four directions. In addition, the recognition of the gestures in MusEd1 is simplified to depend on the gestures' shape, direction and context in MusEd2, thus enabling MusEd2 to accept more sloppy gestures.

The gestures for drawing a minim, drawing a crotchet with a fixed stem direction, changing the stem direction of a note, and changing the duration of a note or a rest are redesigned. There is also an alternative set of double and halve gestures. The rest gesture is reassigned to draw a quaver rest. The single bar-line gesture is redefined to also draw the double bar-line. There are new gestures for crotchet rest, minim rest, and semibreve rest. Finally, the user can choose different notes to draw with the dot gesture. The whole set of gestures in Presto2 can be found in Appendix A. These gestures have been designed to enable fast entry of music into MusEd2.

Other than notes, rests, accidentals, durational dots, beams, bar-lines and double bar-lines, as well as other musical symbols including slurs, ties, accents, staccato, and dynamics are not assigned gestures and cannot be inserted into MusEd2.

Chapter 4

Editing Issues

To edit a document is to modify or improve the contents in that document [micr93, ibmm94]. The editing tools in a system will affect the speed and ease of using an editor [byrd86, farr93, free95]. This chapter reviews editing issues in Presto, the research prototype that consists of the gesture set and the music editor, and how its editing tools can achieve its main objectives, that is to be fast and easy to enter music into the computer. These objectives are described in Section 1.5.

This chapter discusses five issues which are not addressed or are unsatisfactory in Presto1, the gesture set that Anstice [anst96$_a$, anst96$_b$] designed:

- **deletion**

  The user needs a fast and easy gesture to delete a symbol that is inserted accidentally or becomes redundant later.

- **undo**

  Undo is used in a system to reverse an action and return the system to its previous state. Users experiment with the system more easily with an undo function.

- **selection**

  Selection can facilitate group edits in the system, such as copy, cut, paste, delete, and transpose.

- **scrolling**

  When the contents are larger than the window of a system, the system has to provide a way for the user to see the contents that are not in the window. Some way of automatic scrolling is also needed to help musicians enter music smoothly into the system.

- **the sizes of the staff and the cursor**

  The size of the contents in the window affects the legibility of the contents, and the sizes of both the contents and the cursor of the system influence the drawing of the gestures in a pen-based system. The appropriate sizes of the staff and the cursor in Presto will help the user to enter music with ease.

4.1 Deletion

In an editor, deletion erases an unwanted entry that the user has made earlier. One of the more common ways to delete is to first select a symbol or a group of symbols, then press the *delete* key on the keyboard or select *delete* on a menu. Many text and music editors use this method of deletion. They include Mockingbird which is presented in Section 2.2.2, Cantor's editor presented in Section 2.3.1, SSSP editors presented in Section 2.3.2, and the CNMAT system presented in Section 2.3.4. Although this is a familiar way to delete, it involves more than one step in the deletion process. That is, the user's intent is separated into two actions; which command to execute and which symbols to execute the command on. Thus it is not a fast and suitable method to use in a pen system.

Another method of deletion which is more common in pen systems is to use a delete gesture. Some of these systems and their delete gestures are listed in Table 4.1. These gestures consist of one or two strokes. Most of them are straight lines and some are curved lines. Pen gestures can input information in a way that menu-based commands cannot; they convey the information on which command to execute and which symbols to execute the command on at the same time.

The shortest delete gesture in Table 4.1 is the tap gesture in the Schoolchildren system [kimu96]. In this system, users tap the pen to draw a dot gesture on an object to delete that object. Goldberg and Goodisman [gold91] did not assign a specific delete gesture to the text entry system. They wanted the system to imitate pen and paper techniques of

editing text, but found that the action of turning a pencil to its eraser on the other end to rub off unwanted marks is complicated and slow. Instead, they suggested to click the button on the pen barrel and rub off the mark. Users preferred this gesture to turning the pen, because the former is a faster and easier way to erase marks.

4.1.1 Current deletion

The assigned gesture for deleting a symbol in Presto1 is a *scrub* gesture  . This gesture creates problems in deletion for three reasons. It is not natural for musicians to use, difficult for the user to draw, and difficult for the system to recognize.

System	Description	Gesture
Air traffic control system [chat96]	*delete*	
Apple Newton [meye97]	*delete*	
GEdit [kurt91$_a$, kurt91$_b$]	*delete*	
Gesture Mosaic [mosa97]	*delete* or *backspace*	
GO Penpoint [mill$_1$93]	*delete*	 or 
Graffiti [hewl$_1$95]	*delete*	
	backspace	
Network design system [mard93]	*delete*	
Schoolchildren system [kimu96]	*delete*	• tap on object
Text entry system [gold91]	*delete*	click pen button and rub off
Unistrokes [gold93]	*backspace*	
Windows for Pen Computing [ouel95]	*delete*	
	delete word	

Table 4.1: Delete gestures in different pen-based systems.

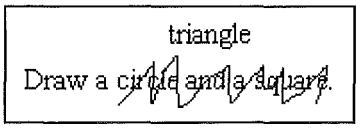


Figure 4.1: Deleting and inserting text by scrubbing.

Figure 4.2: Deleting and inserting musical symbols by scrubbing; this is incorrect.

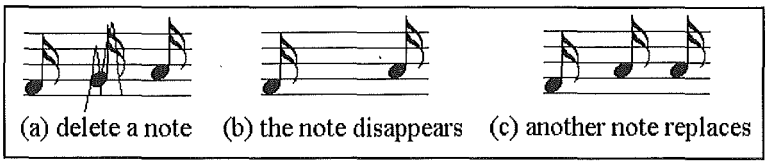


Figure 4.3: Deleting and replacing musical symbols correctly.

Musicians delete musical symbols different to the way writers delete text on paper. Writers draw a *scrub* on the error and write the correction to one side, as shown in Figure 4.1, but it is not natural for musicians to draw a *scrub* on the incorrect symbol and put a correct symbol nearby like in Figure 4.2. This editing shorthand is not allowed in music, because the two-dimensional (horizontal and vertical) spatial position of music notation is important. Musicians correct music notation by rubbing off the error and putting the correct symbol in a correct and appropriate place. An example is in Figure 4.3. Hence, deletion is more complex in music editors than in text editors and it needs a gesture that is natural for musicians to draw.

A new system should be easy and fast to learn. However, users have found that the scrub gesture in Presto1 is difficult to learn and draw. This situation was observed when subjects tried out the system in the survey on beam gesture which is discussed in Section 5.4. Nine out of ten subjects had trouble deleting symbols in MusEd1, the system designed to test Presto1, using the scrub gesture. They had to draw it several times before the system recognized the gesture. Even users who are familiar with the system, including the author, often could not delete successfully the first time. Drawing the

gesture too many times will frustrate the user and make the user feel incompetent, especially because it is one of the first gesture that is needed in experiments and is usually used when the user is already trying to correct an error. As a result, the user may be unwilling to use the system just because the user is not able to get rid of mistakes made by the user or the system.

Users are not the only ones who find the scrub gesture difficult—the system itself has difficulty recognizing the scrub gesture too. Handwriting differs from one person to another, and so do pen gestures. The more complex and the bigger a particular pen gesture is, the more difference there is from one person's gesture to another. This difference causes difficulty in the recognition of the gesture. It has four changes and all the four lines are drawn diagonally. Unlike horizontal and vertical strokes that do not vary much, diagonal lines can range from a flat slope to a steep slope. The problems with diagonal lines in MusEd1 are discussed in Section 3.1. That makes recognizing the scrub gesture more complicated. In addition, the scrub gesture is long and takes more time to draw compared to a dot or a line gesture. For these reasons the scrub gesture is not appropriate, and a shorter and more specific gesture is needed for deletion.

4.1.2 Proposed deletion

One method of deletion is to use the same gestures that insert the different musical symbols but at the same time pressing a button on the pen barrel or an assigned key on the keyboard. For example, the user presses the button or the key and draws a vertical line, which is a bar-line gesture, on a bar-line to delete it or the user presses the button and draws a horizontal line, which is a beam gesture, on a beam to delete that beam. Thus, if users remember the gesture for inserting a symbol, they will also remember how to delete that symbol. However, with this method, the delete command has no universal gesture and a new gesture for group deletion will have to be created. Also, the user has to draw the gesture accurately on the right symbol, especially if symbols such as beams are placed tightly together. In addition, if a symbol is created due to a misinterpreted gesture, the user may find it difficult to redraw that gesture. This means that designing a new gesture for deletion is still inevitable.

The new gesture for deletion can be adopted from any of the delete gestures in Table 4.1, but most of these gestures are long, slow and difficult for users to draw, or difficult for the system to recognize.

Tapping with the pen, that is drawing the dot gesture, in the Schoolchildren system [kimu96] is the shortest gesture and it is already used in Presto1 to draw a crotchet. This gesture can be modified to require pressing the button on the pen barrel or a key on the keyboard for a command that is used often, namely deletion.

In addition to the advantage that it is short, the button-and-tap gesture will not cause accidental deletions because users have to press the button deliberately. Accidental dotting without pressing the button, which results in a crotchet gesture, is a problem that occurs frequently when the user draws gestures in MusEd1. This happens if the user unintentionally lifts the pen at the start of a non-dotting gesture before completing the whole gesture. Eight out of ten subjects drew unintentional crotchets when they were experimenting with MusEd1 in the survey that was conducted for finding out how musicians draw beams. This survey is discussed in Section 5.4.

4.1.3 Implementation

The button-and-tap gesture is implemented in MusEd2 to delete musical symbols. MusEd2 is the system developed from MusEd1 to test Presto2, and Presto2 is the gesture set that the author developed from Presto1. The symbols that can be deleted include notes, beamed notes, rests, accidentals, beams, and bar-lines. The deletion gesture can also change double bar-lines to single bar-lines. A button-and-double-tap gesture can be used to delete whole beams and double bar-lines. The deletion of beams is discussed in Section 5.6 and the deletion of whole beams is discussed in Section 5.6.1. Deletion is also possible if the user presses the *shift* key on the keyboard with the free hand instead of pressing the button on the pen barrel while tapping the pen on a symbol. This may be easier to do, and will be necessary if there is no button on the pen.

4.2 Undo

Undo is an alternative to deletion, but undo has different capabilities because it can reverse almost any immediate action that the user regretted performing and return to the system's previous state [brig$_1$87, dann90, coop95]. Undo is a powerful and important editing tool to include in an editor. It is used in systems to correct mistakes that are made by the user or the system. The user's mistakes may be unintentional, regretted, or mistakes that are caused by the user's misunderstanding of the system [arch84, styn90].

The system makes mistakes when it has implementation errors, or it misclassifies or misrecognizes gestures especially in a pen system [thim90, rubi91].

The user usually undoes a gesture that the system misrecognizes before the user retries that gesture [rubi91]. A quick and easy undo will encourage the user to experiment with the gestures. If the user is new to the system or if the user misunderstands how the system works, a good undo can help and support the user's practice with the system [coop95, mora95], strengthening the interaction between the user and the system [thim90]. Cooper [coop95] states that confidence is the vital psychological contribution that the user gets by being able to undo. "Both the system and the user could be bolder in their actions." [arch84] The system can be automated to a larger extent and the user will be more determined to experiment with the commands in the system.

Thimbleby [thim90] lists six requirements of undo:

- it must be simple to execute;
- it must be general to undo any command;
- it must not be restricted to only certain commands;
- it must be useful;
- it must make limitations reasonable and obvious to the user; and
- some commands should have their own undo commands.

Many systems enable users to execute undo through menus, but that is not desirable in pen systems because pen systems use gestures as the main input method. Menus in pen systems are used for less common commands. Only some pen systems provide gestures to execute undo, while others do not have this command at all. The undo gesture in Windows for Pen Computing is  [ouel95], in the Network design system is  [mard93], and in the Air traffic control system is  [chat96].

There are generally two types of undo: single undo and multiple undo. Single undo can undo only one previous action, while multiple undo can undo a number of actions. Single undo is implemented more frequently in programs than multiple undo, because it is simple. However, it is not useful if the user does not undo an action immediately or if the user has accidentally performed an action such as an innocent click of the mouse before the user can undo, because the action that the user intends to undo is not in the

system's memory anymore. Multiple undo is useful when the user wants to backtrack more than one step, but it needs more memory and it slows down the system [coop95].

4.2.1 Current undo

Undo is not implemented in MusEd1. The absence of this command makes editing music in MusEd1 laborious, especially when music presentation has to be precise. When the user cannot undo a regretted action, editing in the music system becomes frustrating. Users who are learning to use the system can achieve much improvement when they experiment with the system and they are able to undo any actions that they have performed. A redo command which undoes the undo command would make the system even more flexible for users.

4.2.2 Proposed undo

A proposed gesture for the undo command in MusEd2 can be similar to those in other pen systems. However, these gestures contain curved strokes and MusEd2 will have more difficulty recognizing these than straight lines because curved lines varies in different handwriting more than straight lines. An undo gesture itself should not be easily misrecognized because the user will not be able to undo the previous error as well as the undo gesture that is misrecognized.

The button-and-tap gesture is used to delete symbols in MusEd2, as discussed in Section 4.1.2, and it must be placed on a symbol for the command to be executed. If the tap is not on a symbol, nothing will happen in the editor. This situation can be taken advantage of and be assigned as an undo gesture. Thus, drawing the button-and-tap gesture in blank space and not on any musical symbols will execute undo. A double tap in empty space with the button pressed could execute redo. This is fast and simple but it may be misrecognized as deletion or vice versa if the user draws near a symbol.

Another simple and suitable candidate as an undo gesture is a horizontal line. In Presto1, a horizontal line to the right is drawn on notes and rests to add a durational dot, and a line is drawn to the left to remove a dot [anst96$_a$]. There is no action associated with the gesture if the gesture is drawn in empty space, that is, not on a musical symbol. Thus a line to the left can be used as an undo gesture, and a line to the right can be used as a redo gesture. These gestures are suitable and mnemonic, because the left stroke is

similar to a left arrow for going backwards, and the right stroke is similar to a right arrow for going forwards.

4.2.3 Implementation

The undo gesture is a left stroke and the redo gesture is a right stroke; they are implemented in MusEd2 and they can be drawn anywhere in the window except on the staff. Only single undo and single redo are implemented. In other words, the user cannot undo consecutively. This is the same for redo. In addition, the user can redo only if there was an undo just before. The user can undo most commands, including group edits enabled by selection which is discussed in Section 4.3, and discarding the current score to get a new staff. The system cannot undo or redo selections.

Since redo can only be possible if there is an undo just before, the system stores an extra copy of the current state of the staff before it executes undo. If the user next issues a redo, the copied version of the staff is used as the current version.

The implementation of undo is more complicated than redo, because undo has to keep track of the different states of the staff that are affected by many commands. When the user issues a command and before the system executes it, the system stores a copy of the previous state (version one) as well as an extra copy (version two) of the current state of the staff. If the execution of the command is successful and the user next issues an undo, version two of the staff is used as the current version. However, if the execution of the command is unsuccessful and the user issues an undo, version one is used as the current version.

4.3 Selection

Selection is important for users who wish to perform an editing function on more than one object or symbol. Such group edits in music editor include deletion, moving, and transposition. The duration and the direction of the stems of notes can also be changed as a group.

There are two methods that can be used to select objects. The first selection method is the same as that in text editors. That is, the user selects objects (or words in text editors) by dragging over them with a pointing device, such as a mouse or a pen. This is

convenient when the objects, especially languages, are one dimensional. Unistrokes in Section 2.1.1 and Mockingbird in Section 2.2.2 use this method to select objects.

The second method is to enclose the selected objects with a rectangle or an enclosure. The gesture drawing a rectangle is used in Landay's interface design editor [land95], and the enclosing gesture is suggested by Buxton et al. and implemented in *scriva* [buxt79], which is presented in Section 2.3.2. GEdit [kurt91$_a$, kurt91$_b$], the Air traffic control system [chat96], and Tivoli [kurt94$_b$, pede95] also use the enclosing gesture to select objects. In these editors, the objects of music or graphical display are not fixed in their two dimensions; the square and the enclosing gestures are very suitable for such objects.

Objects in the selection can be excluded by drawing another enclosure or square around them, as illustrated in Figure 2.20. If these gestures are used to select the same objects, the rectangle gesture is faster because it only needs a diagonal line to indicate two diagonal corners of the rectangle. However, the enclosing gesture is easier to include some objects and exclude other objects at the same time because it does not consist of straight lines. Thus, the user can draw only one enclosing gesture, as shown in Figure 4.4a, to select objects in awkward positions which will need two or more square gestures to include and exclude them, as illustrated in Figure 4.4b and Figure 4.4c.

4.3.1 Current selection

There is no selection gesture or command in Presto1 and MusEd1, and group editing is not possible. Selection is needed in Presto especially for expert users when they cut and paste groups of notes and transpose them. In this way, it will be faster and easier than drawing individual notes repetitively.

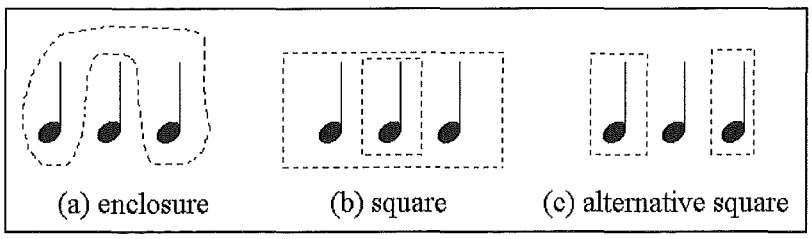


Figure 4.4: Different selection gestures.

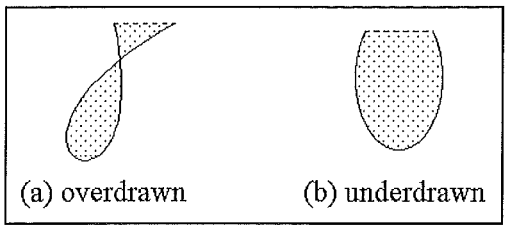


Figure 4.5: The selection gesture.

4.3.2 Proposed selection

Dragging symbols as in text editors is not suitable for selection in music, because it is two dimensional. The selection gesture cannot be a straight line for drawing a square gesture, since it will conflict with other gestures like the beam, as well as add and remove dot gestures. Appendix A lists these gestures. Thus, the enclosing gesture is a better option for selection than the square gesture.

4.3.3 Implementation

Since the recognizer in Presto2 is simple and uses straight lines, the enclosing gesture cannot be recognized accurately. It can even misrecognized as other gestures in Presto2, especially those that consists of two directions such as the tail-down gesture and the semibreve gesture. Therefore, in order for the enclosing gesture to be recognized, it must be differentiated from other gestures in Presto2. The button on the pen barrel or the *shift* key on the keyboard can be pressed to do selection. The only other gesture that uses the button is the delete gesture. Thus, if the gesture is not a dot, then the gesture is selection. If a better recognizer is used, the user does not need to press to select symbols.

The enclosing gesture may be overdrawn or underdrawn like in Figure 4.5. In these situations, a straight line will join the ends of the gesture and enclose the selection. If the gesture includes symbols that are already selected, they will become unselected.

All selected symbols will change to light blue. Light blue is chosen, because light blue is displayed as gray on the monochrome tablet used for Presto and is clearly distinguished from black. The user can select accidentals, rests, notes by drawing around their heads like in Figure 4.6, and bar-lines and beams by including part of the symbols in the enclosure. After selection, the user can issue several commands to the selection by

drawing the gesture of the commands on any selected symbols. To unselect all the symbols, the user draws another enclosure in any area that the staff is not in.

4.4 Scrolling

Scrolling is a familiar feature found in editors. Unless the system is like Cantor's editor, presented in Section 2.3.1, that does not use scrolling and four scopes of screen are needed, scrolling is needed for viewing large amounts of contents in a relatively small window on the screen [schw83, shne92]. Moreover, it is not practical to have multiple screens or to increase the resolution of the screen to get more information [norm$_2$86]. Issues affected by the actual size of the contents will be discussed in Section 4.5.

The most common mechanism in scrolling, that is, to move the window over the contents, is the scrollbar. Shneiderman [shne92] lists five features of a scrollbar:

- it should allow for small or large movements;
- it should be able to move the window incrementally or jump to a destination;
- it must produce feedback on the results;
- it should have up and down arrows for the user to click on to move the window by a small motion; and
- when the up or down arrow is clicked and held on, the window should scroll continuously and smoothly.

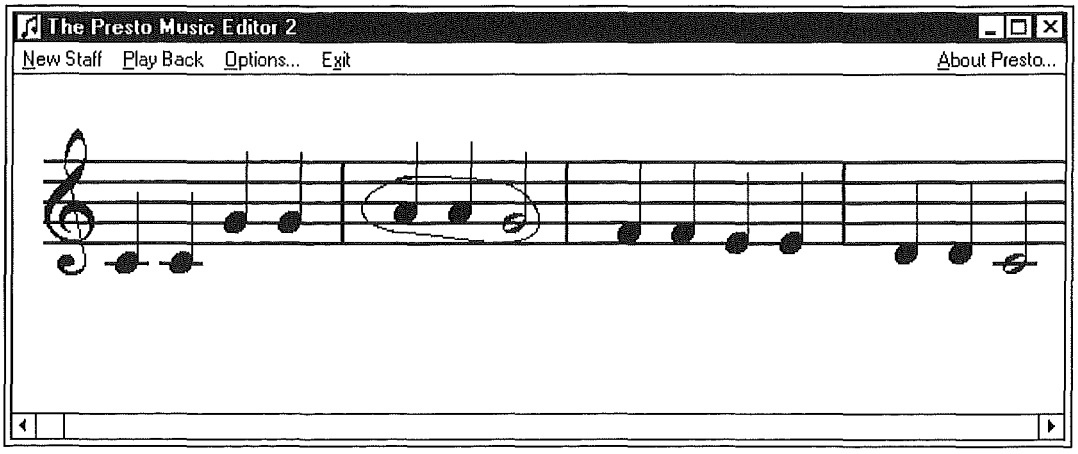


Figure 4.6: Selection gesture in MusEd2.

Pen systems can use other types of scrolling mechanism. Perkins et al. [perk95] designed a product development design editor to scroll the contents instead of the window. If the user moves the pen to the edge of the tablet, the cursor changes into the shape of a hand. Next, if the user gestures a sliding motion with the pen, the image on the tablet will scroll, just like moving paper on a desk.

In addition to scrolling mechanisms, modern text editors have an automatic scrolling function that prevents text from flowing off the screen while the user is typing. Automatic scrolling also relieves users from scrolling manually while they are entering text continuously.

4.4.1 Current scrolling

A horizontal scrollbar is implemented in MusEd1 since the score manuscript is often larger than the window. However, there are two problems with using the scrollbar in the editor.

Firstly, the position of this scrollbar is not appropriate; the manuscript is at the top of the window and the scrollbar is placed at the bottom of the window. When the user is editing music in random places of the manuscript, the user has to continuously move the pen up to the top of the window to edit and down to the bottom of the window to scroll the window. This process slows down the speed of editing music in the editor.

Secondly, if the user enters music continuously, the user has to scroll manually when the input reaches the edge of the window. In other words, the user has to move the pen to the bottom of the window to scroll the manuscript further, and move back up to the manuscript to continue entering music. If the user does not scroll the window, the musical symbols will flow off the window and the user will not be able to keep track of what and where symbols are entered. It is also distracting for the user to move away from the manuscript when the user is composing or editing music.

4.4.2 Proposed scrolling

In Section 1.5, one of the objectives of Presto is to enable fast entry and edit of music, therefore the editor should allow the user to manipulate the window in an area near to the score. Rather than operating on the upper or lower border of the window, the left and right edges of the window can be used as the window's holder.

If the user wants to move the window to the right, the user can move the pen to the right edge of the window and the cursor will follow the pen and change into a hand figure. At this point the user can hold the pen down, the cursor will change into a grabbing figure and the window will slide to the right of the score. Direct feedback of the position on the score is needed through animation of the sliding window. Feedback is discussed in Chapter 6. In this way, the user does not need to move the pen, the user just holds the pen down. The scrollbar in MusEd1 can remain in MusEd2 to compare the efficiency of the two scrolling tools.

MusEd2 also needs an automatic scrolling function to prevent the musical symbols from flowing off the screen if the user does not scroll, and to assist the user in composing so that the user does not need to scroll manually. Automatic scrolling assumes that the next action that the user is going to perform is the position next to the symbol that the user has just entered or edited and the window will keep the position of this next action in view.

When should the window scroll automatically? The window can make sure that the bar that the user is editing is placed in the middle of the window. However, if the user has not drawn a bar-line, or if the bar that the user is editing contains so many symbols that the window is unable to contain the whole bar, the automatic scrolling may not operate efficiently. The window can also scroll automatically depending on the precise symbol that the user is entering or editing. When the user edits until the last few symbols in the window, for example, the third last symbol, the window will automatically scroll that symbol to the middle of the window. Animation of scrolling can be used as feedback to the user. Chapter 6 discusses issues on feedback.

A subject in the survey in Section 5.4 tried to scroll with her finger. If the tablet can accept pen input as well as finger input (touch sensitive), the right or left hand can hold the pen and use it to draw gestures while a finger on the other hand can scroll the window when needed. Two-handed input has proved to be an effective way of human-computer interaction [buxt86$_b$]. Since the tablet that Presto uses does not accept finger input, the hand that is free can use the mouse or press on a key on the keyboard to scroll the window.

4.4.3 Implementation

In addition to the scrollbars, the left and right arrow keys on the keyboard are implemented in MusEd2 for manual scrolling. The *home* and *end* keys are implemented to jump to the beginning and the end of the score respectively. Manual scrolling can also be done using the mouse. If the user presses and holds the left button on the mouse and moves the mouse left or right, the cursor will change to a two-ended arrow to indicate scrolling, and the window will scroll over the score. A mouse button has to be pressed to activate manual scrolling, because the mouse can often be moved accidentally; if the score can be scrolled without the mouse button, the window will scroll as often as the mouse moves. The user can use the free hand to press the key or the mouse while the other hand is still holding the pen.

One restriction in MusEd2 is that the user cannot use the mouse and the pen at the same time, because they share the same cursor and their different movements will conflict. The pen and the mouse can be designed to have their own cursor to avoid this problem.

MusEd2 will also automatically scroll the window when the user draws gesture on the left or right edge of the score. The window will scroll and put the point where the gesture is in the middle of the window.

4.5 Sizes of staff and cursor

The size of the contents of a window, such as the font size of characters in a text editor, is important to the user. If the font size is too small the user may have difficulty reading the document. If the font size is too big, the user may have to scroll the window unnecessarily to read the whole document. Thus the size of the contents in a window affects reading, entering, and editing the document.

The size of the contents is equally important in the window of a music editor. The size of the musical symbols depends on the size of the staff. If the sizes of the staff and the symbols are too big, the window will not be able to show many symbols at one time. Users will also draw larger and slower gestures on a big staff. If sizes of the staff and the symbols are too small, the user will have difficulty reading and comprehending the music score. Drawing gestures and positioning the musical symbols will also be inaccurate and slow on a small staff due to the requirement of fine motor movements of the hand.

Similarly, if the pen cursor is too big, it will prevent learners from observing and learning the new pen gestures during a demonstration of the system, and it will also obstruct the view of some musical symbols in the music score. If the pen cursor is too small, the user will have difficulty finding it. Since a pen system uses direct manipulation and direct screen contact, it does not need the same feedback from the cursor like in a mouse-based system, because the pen itself will still provide the information about the position of input.

There is not much research about the size of a staff in the literature on music editors. Mockingbird which is presented in Section 2.2.2 has seven staves per screen, Cantor's editor presented in Section 2.3.1 has four staves per screen, and SSSP editors presented in Section 2.3.2 have two staves per screen. It is hard to judge the sizes of the staves in these editors.

Research on the font size of characters on television and computer screens can be found in papers on human factors. A standard computer screen has 24 lines and 80 columns, which is a third of a sheet of paper [fins82]. The optimum character height is between 3.96 millimetres (11 points) and 4.79 millimetres (13.5 points) [gidd72, snyd79], and the maximum viewing distance is 1.5 metres [snyd79]. Any character height taller than 4.79 millimetres (13.5 points) or any distance shorter than 1.5 metres will not improve legibility [snyd79]. Pastoor et al. [past83] found that a 9 by 13 matrix size is the most suitable at a viewing distance of six screen heights.

4.5.1 Current sizes

The size of the staff in MusEd1 may be too big, because only a small number (12 to 17) of musical symbols can be shown in the window at one time. The pen cursor in MusEd1 may be too big or unnecessary, because it obstructs subjects in the beam survey, which is discussed in Section 5.4, from viewing the pen ink and learning new gestures during the demonstration.

4.5.2 Proposed sizes

An experiment with different sizes of the staff and the cursor is needed to find the appropriate ones for MusEd2. However, different users may prefer different sizes. Sizes are judged by the number of pixels per staff height. Users can choose from a menu that provides various staff sizes ranging from small to large. A menu is preferred over pen

gestures because changing the staff size is an uncommon command, and users will change the size of the staff once or twice with every new staff. A large staff can be used for music or system demonstration, while a small staff can help more experienced users manage the overall view of the music score. In a production system, users would also like to switch quickly and easily between large and small views.

The size of the pen cursor can also change according to the size of the staff, but the different sizes of the cursor may not differ as much as the staff size. Similarly to the staff size, users may have different preferences to the size of the cursor. Another solution is to make the cursor translucent. In this way, the cursor is not obstructing the user's view of the contents of the window but the user will still be able to find the cursor. The third way to solve the problem is to remove the cursor altogether. However, if the user uses the mouse to draw the gestures, the user will still need the cursor for the indirect manipulation of the mouse. The best way is to provide a menu to turn the cursor on or off.

4.5.3 Implementation

An option of three staff sizes is implemented into the main menu of MusEd2. The medium size (48 points) of the staff is the size used in MusEd1 and the default size in MusEd2, as shown in Figure 4.7. The small size (38 points) is 80 percent of the medium size and it allows 16 to 24 musical symbols to be shown in the window, as shown in Figure 4.8. The large size (58 points) is 120 percent of the medium size and the window can contain 10 to 14 symbols, as illustrated in Figure 4.9. Another issue arises when this option is implemented into MusEd2: if the user starts another new staff, should the size of this staff be the default size (medium) or be the size that the user has chosen? Since the user has already chosen a staff size, the new staff size should remain as it is.

The main menu has another option which can turn the cursor on or off, as shown in Figure 4.10; the default is on because it will be useful to let the user check if the pen and the tablet is working properly when the user starts up the system.

4.6 Summary

This chapter has discussed editing issues in Presto; they are deletion, undo, selection, scrolling, and staff and cursor sizes. Deletion can be made on most of the musical symbols in MusEd2 using the button-and-tap gesture. The *shift* key can also be used

instead of the button on the pen barrel. Single undo and redo are implemented in MusEd2 to enable users to experiment with the gestures in Presto2.

An enclosing gesture around musical symbols in MusEd2 selects these symbols and lets users execute commands on the selection. Manual scrolling is possible by using the scrollbars in the window, the arrow keys on the keyboard, and the mouse. MusEd2 also automatically scrolls the window if the user gestures on the edge of the window.

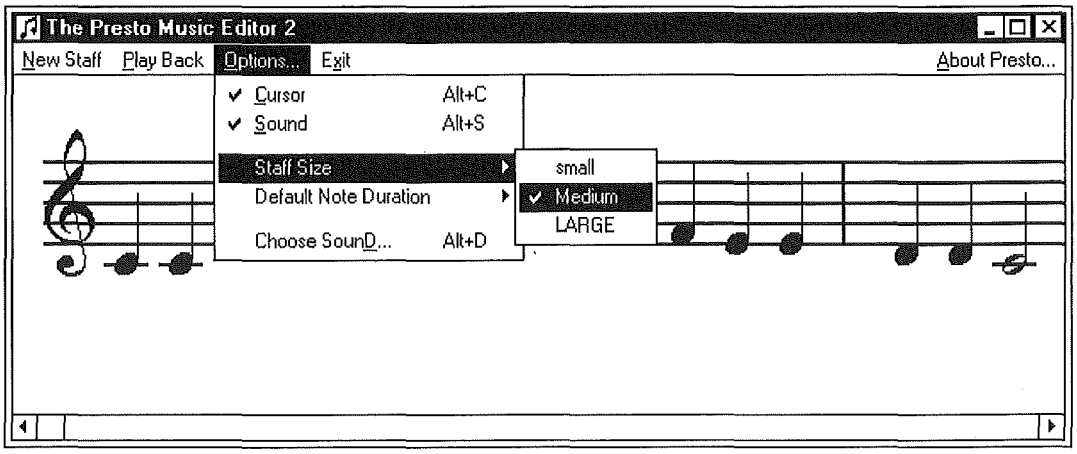


Figure 4.7: Medium size of the staff in MusEd2.

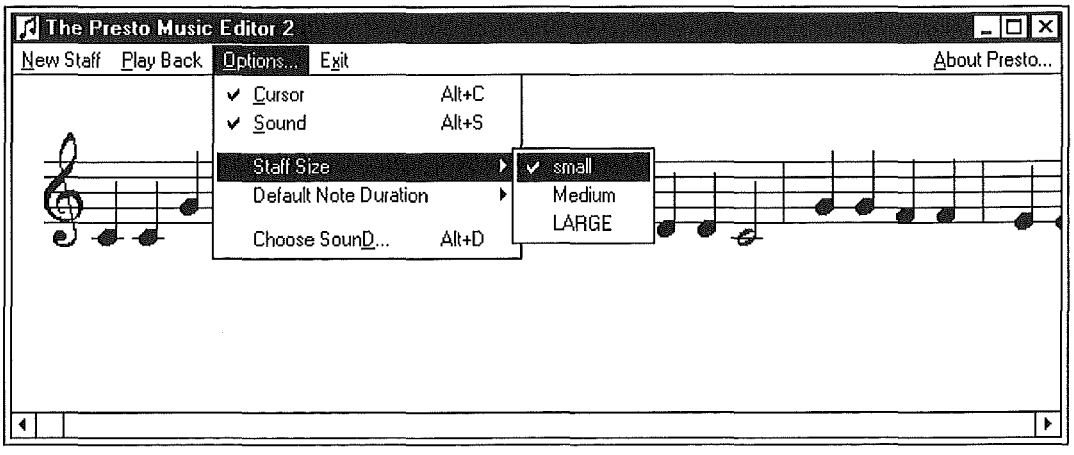


Figure 4.8: Small size of the staff in MusEd2.

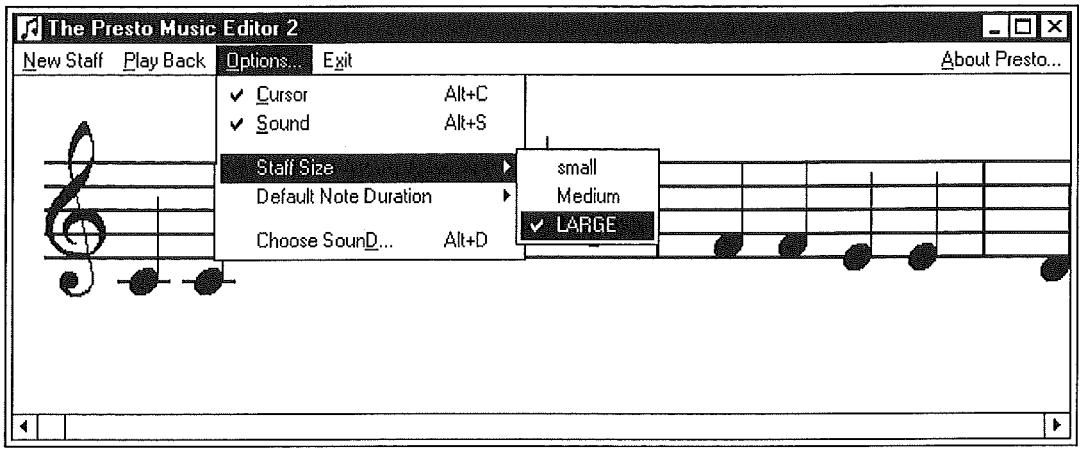


Figure 4.9: Large size of the staff in MusEd2.

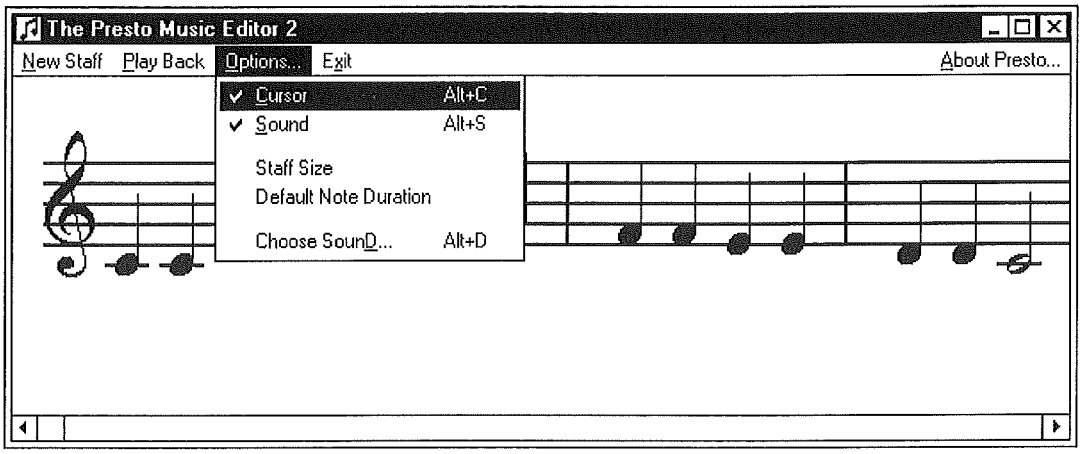


Figure 4.10: Cursor option in MusEd2.

Users can experiment with three sizes of the staff in MusEd2 and select the appropriate size according to their need and comfort. Users can also switch the pen cursor off if it interferes with their use of the system. These editing features implemented in MusEd2 will assist users to edit music faster.

Chapter 5

Beams

Musicians often use beams to show implied accents on notes and to group notes in a score. An implied accent is usually placed on the first note in a group with beams according to the beat of the score, like in Figure 5.1. Beaming also presents music more clearly than using a whole string of individual notes with flags, as shown in Figure 5.2. For each note in the beamed group, the number of beams is equivalent to the number of flags the note has. Beams are also used to match notes with syllables in scores for vocal music, as illustrated in Figure 5.3. However, the process of drawing beams and deciding on the placement of those beams on notes is surprisingly difficult when beamed groups become more complex.

This chapter first looks at some common music notation rules on beaming notes as well as the issues on the gesture for drawing beams in Presto1, the gesture set that Anstice [anst96$_a$, anst96$_b$] designed. Next, a paradigm for the beam gesture to solve these problems is presented. In addition, a survey performed to find out how musicians draw beams is reported, and the paradigm is tested on some examples. Methods for deleting beams are also considered, because errors are inevitable. Finally, this chapter looks at how the beam gesture includes notes in beaming, possible data structures for beams, and the implementation of the beam gesture. Some examples presented in this chapter are not conventional rhythmic grouping, they only illustrate the points that are discussed.

Some exceptional beaming and modern practices will not be dealt with in this research. Figure 5.4 illustrates some examples of such beamed groups. Other issues relating to beams not included in this research are the degree of beam slants and the positions of beams on a staff. These are discussed in more detail by Ross [ross70] and Brodhead [brod96$_a$, brod96$_b$].

5.1 Rules about beams

Common Music Notation (CMN) is the *common language* in music and the standard for presenting music. Although musicians may invent their own notation, most musicians follow CMN because other musicians can understand the notation. There are two types of rules in CMN for drawing beams in music: one is for beams in general and one is specifically for broken beams. A broken beam is a beam that belongs to only one note and not to others in the whole beamed group. When rules on beams are applied, some problems will arise due to complications that occur in practice.

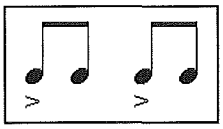


Figure 5.1: Accents on beamed notes.

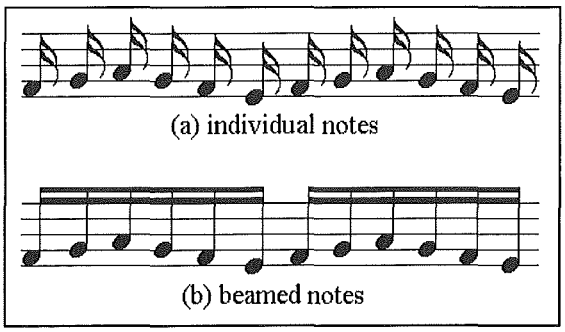


Figure 5.2: Two groups of semiquavers.

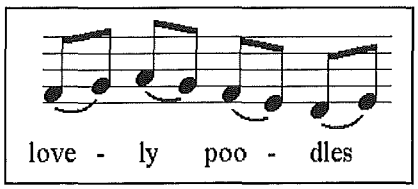


Figure 5.3: Beamed notes with words.

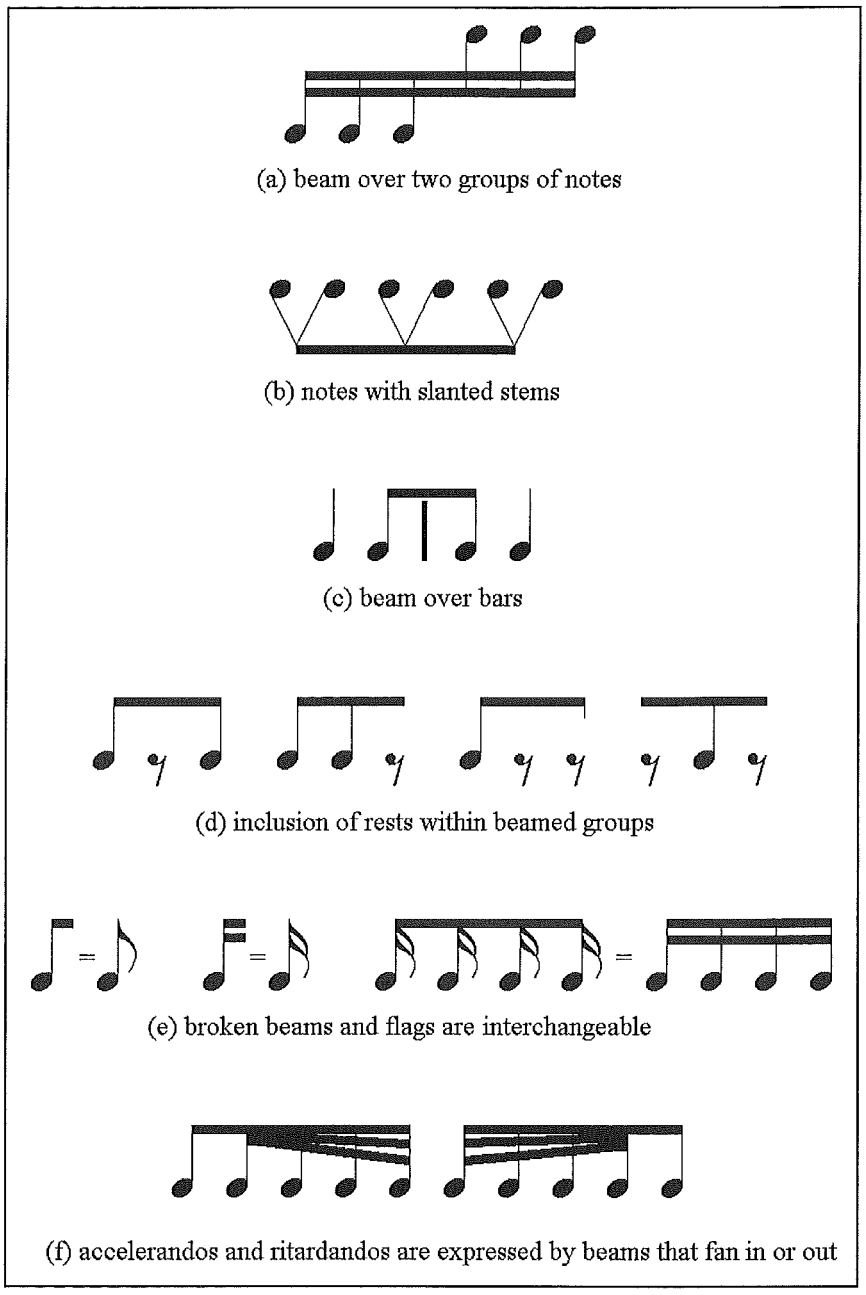


Figure 5.4: Exceptional beaming ([read69], pp. 90-94).

5.1.1 General rule

The problems of beam placement are mainly caused by the most general rule in CMN for beaming: beams must "demonstrate the metrical and rhythmic divisions within the measure." [read69] This means that the notes with beams on them must always be

grouped according to the measure rhythm of the score. Figure 5.5a shows the correct grouping of some beamed notes. The group in Figure 5.5b are incorrect because the notes are not grouped according to the beat.

5.1.2 Rule on broken beams

The general rule in Section 5.1.1 also applies to broken beams, that is, a broken beam must reflect the beat of the beamed group. The short horizontal line on the second note of Figure 5.6a is an example of a broken beam that points to the left. There are three other rules in CMN that restrict the direction that broken beams point. First, a broken beam is "always placed inside the group." [read69] Second, a broken beam of the third or lower level must point in the same direction as the beams of the higher level on the same note. Third, there should not be broken beams of the same level pointing towards each other on two consecutive notes.

The first rule on broken beams state that a broken beam must always be positioned within the beamed group and according to the beat. In other words, if the first note on the left of a beamed group has a broken beam, this broken beam must point to the right. If the first note from the right has a broken beam, it must point to the left. Figure 5.6a shows the correct direction of the broken beam of the second note, while Figure 5.6b shows a broken beam that points incorrectly outside the beamed group.

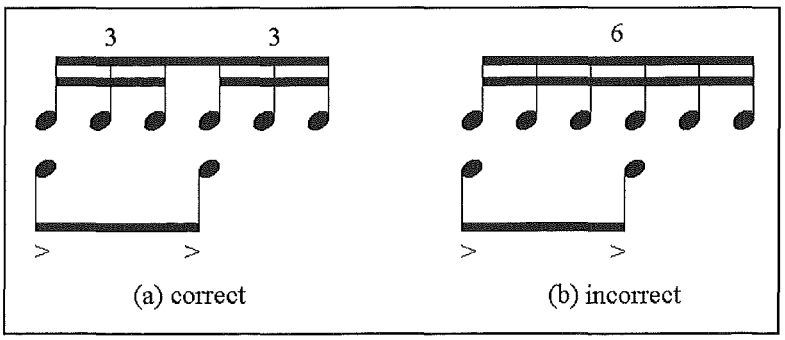


Figure 5.5: Beamed notes and their rhythm ([read69], p. 84).

The situation differs if the broken beam is neither on the first or the last note. According to the rule in Section 5.1.1, the broken beam must point in the direction such that the whole beamed group reflects the group's rhythmic measure. In this case, the broken beam on a note should point to the side such that this note and its neighbouring note or notes on that side forms a beat. For example, the group of notes in Figure 5.7a is correct, but the direction that the broken beam is pointing to in Figure 5.7b is incorrect because the second and the third note together do not form a beat.

Thus, it may seem that a broken beam should point on the same side as the neighbouring note that has one or more durational dots, as in Figure 5.7a. However, this solution does not capture two cases. One case is where both the left and right neighbouring notes have durational dots. In Figure 5.8a, the broken beam on the second note points correctly to the left, but Figure 5.8b shows that it is incorrect for that broken beam to point to the right because the first note does not form a beat by itself. The other case is when neither the left or right neighbouring notes have durational dots. In Figure 5.9a, the broken beam on the third note points to the right to match the last note, which also has a broken beam, according to the rhythm. Pointing that broken beam to the left, shown in Figure 5.9b, or connecting it with the broken beam on the second note to form a complete beam, shown in Figure 5.9c, is incorrect. It is more difficult to solve these two cases, and in the interest of flexibility, it is best decided by the user.

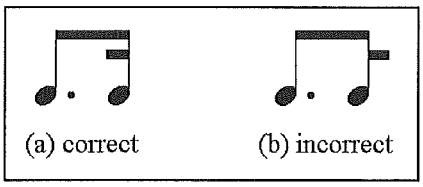


Figure 5.6: A broken beam ([read69], p. 84).

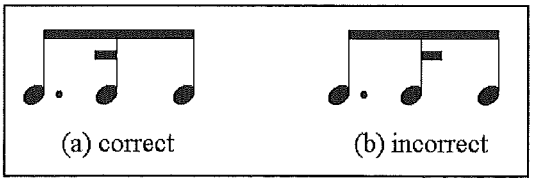


Figure 5.7: A broken beam on the middle note of a beamed group ([read69], p. 85).

The second rule on broken beams applies if there are more than one broken beam on a note. The first secondary beam may be a broken beam or a complete beam over two or more notes. A secondary beam is a beam in the second or lower level. A complete beam is a beam that has both ends touching the stems of notes. In Figure 5.10a, the second broken beam on the second note is below another broken beam, and it points to the same side as that beam. Pointing that second broken beam to the other side is incorrect, as illustrated in Figure 5.10b. Similarly in Figure 5.11a, the broken beam on the second note is below a complete secondary beam. It is incorrect to point that broken beam to the other side of the stem, as illustrated in Figure 5.11b.

The third rule does not allow broken beams of the same level on two consecutive notes to point towards each other, as shown in Figure 5.12b. Instead, a complete beam can replace the two broken beams and have the same effect, shown in Figure 5.12a.

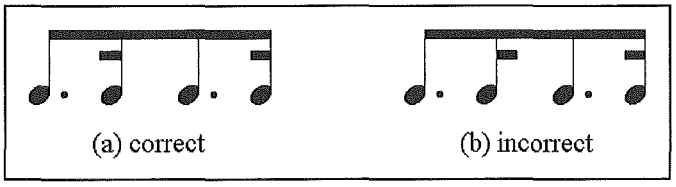


Figure 5.8: The second note has two dotted neighbouring notes.

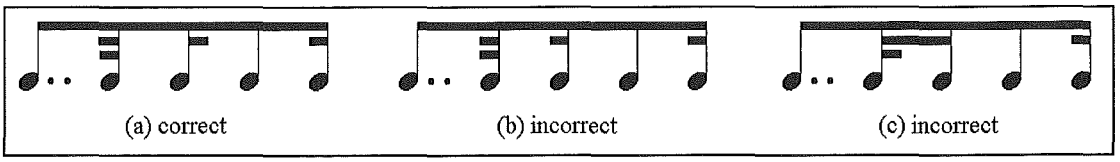


Figure 5.9: The third note does not have dotted neighbouring notes ([read69], p. 85).

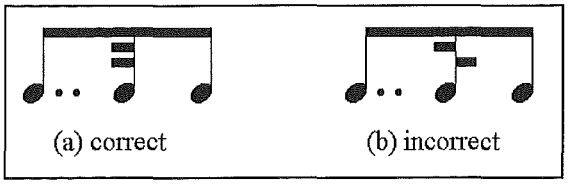


Figure 5.10: A broken beam below another broken beam.

Thus there are four rules that govern beams in CMN. First, the beamed group must reflect its rhythm. Second, broken beams must be placed within the beamed group. Third, if there are more than one broken beam on a note, all the broken beams must point to the same direction. Last, broken beams on two consecutive notes must not point towards each other.

5.2 Issues of the beam gesture

In Presto, which is the research model consisting of the gesture set and the music editor, the beam gesture is a straight line drawn from left to right or vice versa over the stem or stems of one or more notes. Presto will put a beam over the note or notes that the gesture crosses. The notes in the beam gesture must be crotchets or notes of lesser duration. If there are notes that have a longer duration than a crotchet, Presto will treat that gesture as an error. The user can erase or change beams by deleting them. This is discussed in Section 5.6.

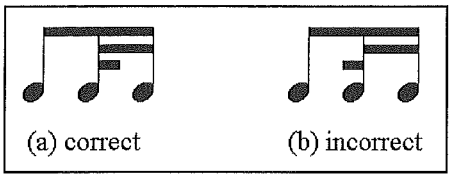


Figure 5.11: A broken beam below a complete secondary beam.

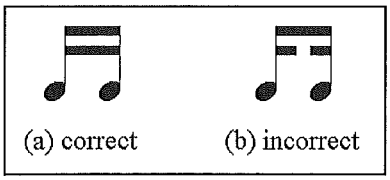


Figure 5.12: Two broken beams pointing towards each other.

There are basically four purposes for drawing beams over a group of notes:

- add a new beam to the notes, as shown in Figure 5.13a;
- connect two beams into one, shown in Figure 5.13b;
- extend a beam over other notes that are not beamed, shown in Figure 5.13c; and
- halve the duration of a note in a beamed group using a broken beam, shown in Figure 5.13d.

This section discusses the issues that are encountered when the beam gesture is used for each of these four purposes. These issues arise mainly because there are many representations of beamed groups and the system has to determine which representation the user wants. The issues of adding and deleting notes in a beamed group are also considered.

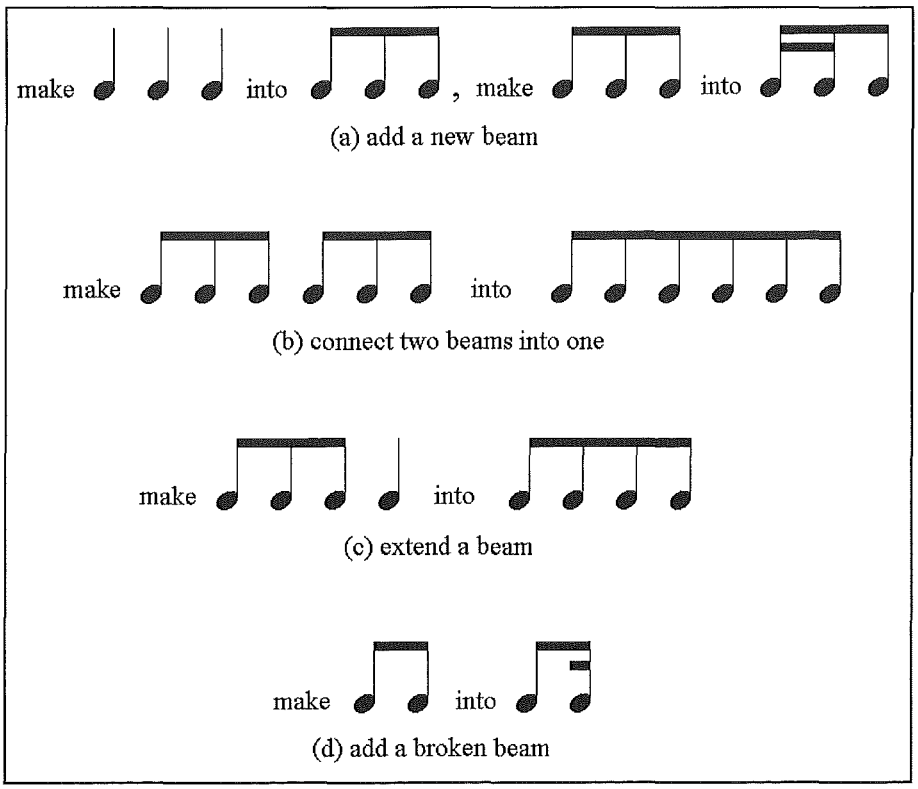


Figure 5.13: Four purposes for drawing beams.

5.2.1 Adding a new beam

Two issues arise when a user adds a new beam to a group of notes, which is illustrated in Figure 5.13a. From the definition of the beam gesture in Presto1, all the notes that are crossed by the gesture are included in the beam. To be specific, the left end of the beam gesture should be at the first note to be beamed and the right end should be at the last note, as illustrated in Figure 5.14. The left ">" represents the left end and the right ">" represents the right end in the figures in this chapter. In Figure 5.14a, the duration of the crotchets in the gesture are halved and a beam is added. However, the question is whether drawing a beam gesture over quavers as in Figure 5.14b, or semiquavers as in Figure 5.14c, halves the notes; the results are different.

Another issue is when the user adds a beam to individual semiquavers or notes with shorter duration as in Figure 5.15, whether the secondary beams of that group should be one beam or should consist of several beams. If it becomes one beam, the user will need to delete parts of the beam to get separate secondary beams. Deleting parts of a beam is discussed in Section 5.6. If the secondary beam becomes a few separate beams, they can be connected if required.

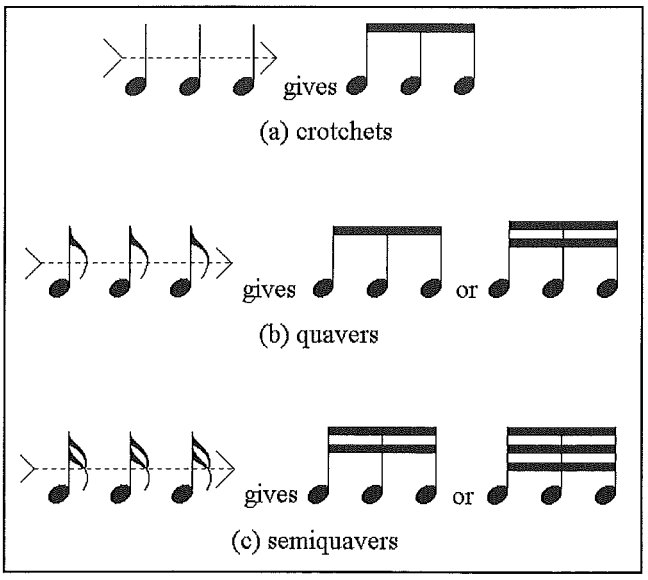


Figure 5.14: Add a new beam.

5.2.2 Connecting multiple beamed groups

There are three issues when a user connects two beamed groups, which is shown in Figure 5.13b. The first issue is at which note the user should put the left and the right ends of the beam gesture in the beamed groups. The left and the right ends point at either the first or the last note of a group with two notes. A group with more than two notes provide three possible places to put the ends of the gesture. The ends can be at the first note, at the last note, or within the beamed group. For an end to be within the beamed group, it must be at any note between the first and the last notes of the group. With three possibilities to position two ends of the gesture, nine combinations are produced to connect two beamed groups into one, as shown in Figure 5.16. The other two issues in this subsection will use the beam gesture in Figure 5.16g for demonstration.

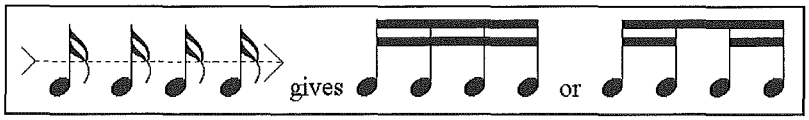


Figure 5.15: Add two beams in one gesture.

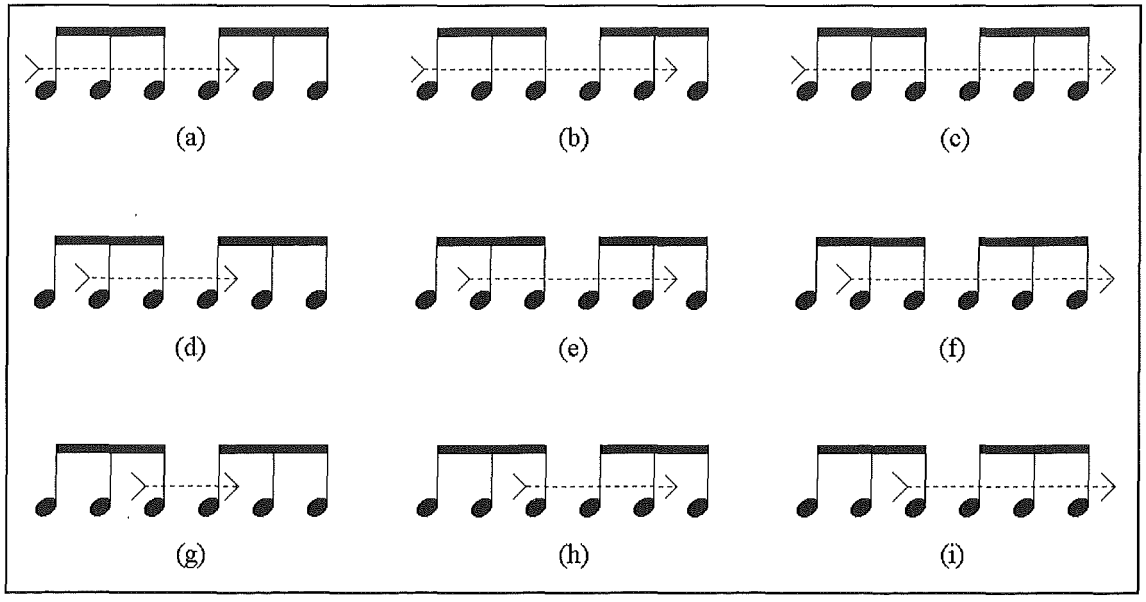


Figure 5.16: Nine combinations to connect two beamed groups.

The second issue arises when the user connects three beamed groups with only one beam gesture. Should this gesture, illustrated in Figure 5.17, connect all three beamed groups into one group, or should it connect and then add an extra beam on the two middle notes of the group? The final issue happens when the user connects two beamed groups that have secondary beams, which is illustrated in Figure 5.18. The user may want to connect only one beam of the groups or to connect all the beams.

5.2.3 Extending a beamed group

A beamed group sometimes needs to be enlarged by extending the beam longer from the left end, the right end or both ends to include individual notes that are next to the group, as shown in Figure 5.13c. When only one end of a beamed group is extended, this end points at the individual notes while the other end points at a beamed group. When a user extends a beamed group, three issues arise.

The first issue is similar to that when the user connects two beamed groups. This happens only to the end of the beam gesture that points at a beamed group. This end has three possible placement in the group; it can point at either the first note, the middle notes, or the last note of the group, as illustrated in Figure 5.19.

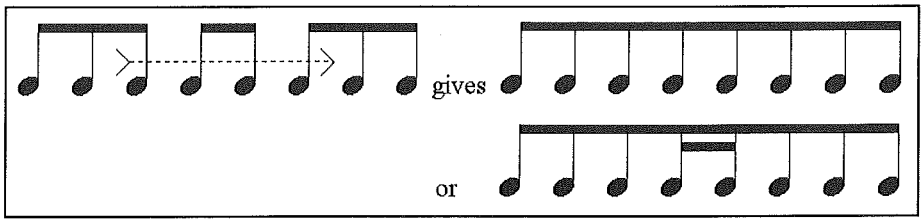


Figure 5.17: Extend a beam over three beamed groups.

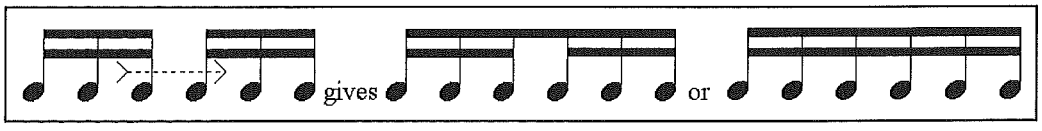


Figure 5.18: Connect two beamed groups with secondary beams.

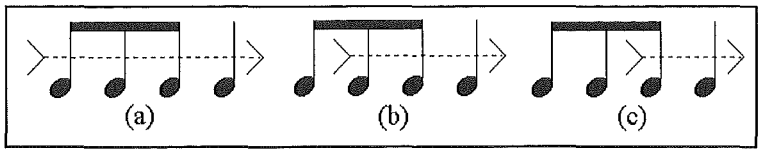


Figure 5.19: Three ways to extend a beamed group.

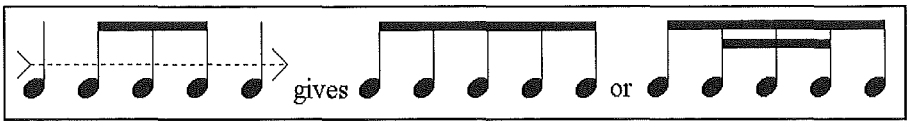


Figure 5.20: Extend a beamed group.

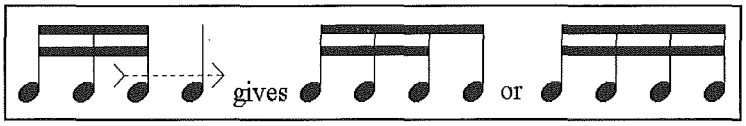


Figure 5.21: Extend a beamed group with secondary beams.

Issue two arises when the user extends a beamed group both ways by drawing one beam gesture. In Figure 5.20, should the result be only one beam on all the notes, or should it add another beam on the middle three notes? The last issue is similar to that of connecting beamed groups. When the user draws the beam gesture in Figure 5.19c to extend a beamed group with secondary beams, as in Figure 5.21, the user may want to extend only one beam or all the beams over the individual notes.

5.2.4 Adding and editing broken beams

If the user wants to halve the duration of only one note in a beamed group, the user has to add a broken beam to that note, as shown in Figure 5.13d. The gesture to add a broken beam is the same as the beam gesture, but the stroke is much shorter. Apart from the rules that broken beams must be within the group and all the broken beams of a note must point to the same side, the decision of left or right placement on the stem of the note is generally left to the user. Thus, one issue with drawing broken beams is how the user should express the decision to point a broken beam to the left or to the right. Another issue is if the user draws a beam gesture over a beamed group that has a broken beam, will the broken beam remain in the group, will the beam gesture extend the

broken beam as in Figure 5.22a, or will the gesture be connecting multiple broken beams as in Figure 5.22b?

5.2.5 Adding and deleting notes in a beamed group

Besides beaming, the user can also make changes to a beamed group by adding or deleting notes in the group. In Presto2, the gesture set that the author developed from Presto1, the user can draw a dot to insert a note in a beamed group and use the delete gesture, which is discussed in Section 4.1, to delete a note in the group.

There is one issue to consider when the user deletes a note in a beamed group: would the neighbouring notes of the deleted note remain with same duration or different duration as in Figure 5.23? The small white circle O in the figure represents the delete gesture.

5.3 Paradigm for the beam gesture

The issues stated in Section 5.2 show that the beam gesture needs a simple paradigm that controls the four needs of drawing beams, so that users can predict what the result of their beam gesture will be and easily determine which gesture to use to get a desired result. There are two rules that the beam gesture could follow; which we call the *logical* rule and the *visual* rule. The logical rule halves the notes that are crossed in the beam gesture, while the visual rule adds a beam in the gaps between the notes that are crossed. The user can further delete some beams to get the desired results. Deletion of beams is discussed in Section 5.6. This paradigm is not well-defined in Presto1, but is close to the logical rule.

The beam gesture can follow a logical rule that is based on the duration of the notes when a group of notes is crossed. The rule halves each note in the group and puts the appropriate number of beams, according to the duration of each note, in the gap between every two notes in that group. Figure 5.24a illustrates a beam gesture using the logical rule. This rule is easy to use on simple beamed groups, but it has problems with more complicated beamed groups because the user needs to draw other gestures to specify rhythm groupings and broken beams. Without these extra gestures, the results of the beam gesture will be incorrect like those in Figure 5.5b and Figure 5.9c. Users will also find the logical rule hard to use.

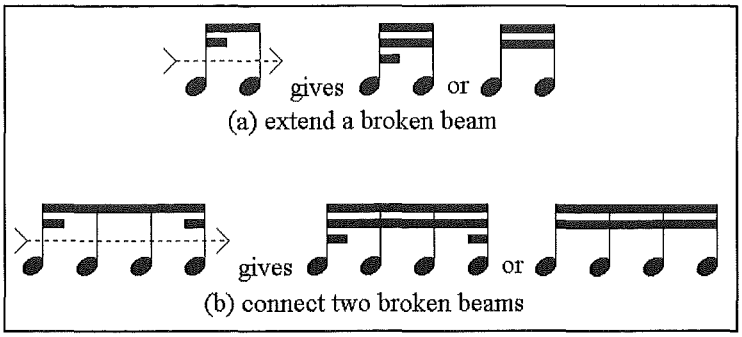


Figure 5.22: Edit broken beams into complete beams.

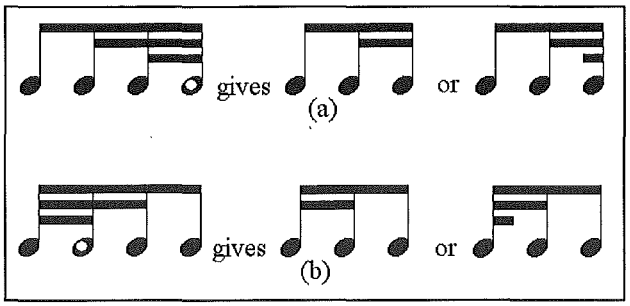


Figure 5.23: Delete a note from a beamed group.

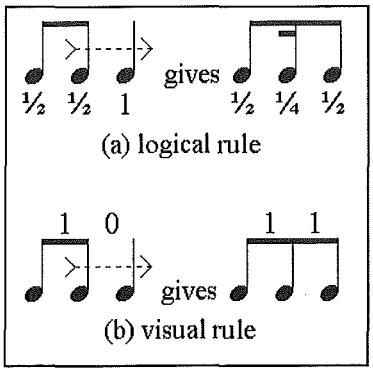


Figure 5.24: Beam gesture using two different rules.

Another rule that the beam gesture can apply is the visual rule. The visual rule depends on the number of beams that a gap between two notes has. A gap between two notes is the empty space that exists between two consecutive notes. The visual rule adds an extra beam in every gap between two notes that the gesture crosses. In other words, if there is

one beam in the gap between two notes, the rule will add one more beam and the gap will have two beams. Figure 5.24b demonstrates the beam gesture that applies the visual rule to the same beamed group in Figure 5.24a which uses the logical rule.

The logical rule affects the stem of the notes and the visual rule affects the gaps between the notes to determine beaming. Either of these rules can be used to draw the beam gesture, but the best rule chosen as the paradigm must be predictable and give the desired results quickly. Deletion of beams, which is discussed in Section 5.6, may be needed in some situations. Section 5.4 describes a beam survey conducted to find out how musicians draw beams and Section 5.5 demonstrates some examples that use the visual rule and compares them to the same examples that use the logical rule. The visual rule seems to be the preferred method in the beam survey, and it also proves to be better than the logical rule as the paradigm for the beam gesture in the examples.

5.4 User survey

A survey on the beam gesture was conducted to observe the way musicians would like to draw beams on groups of notes on Presto using the beam gesture. The aim of this survey is to test the paradigm of the beam gesture formed in Section 5.3.

Ten paid subjects participated in the survey. They were students from the School of Music at the University of Canterbury; three of them were male, seven were female, and all were right-handed. None of the subjects had used a pen-based computer before. The subjects were first introduced to Presto1 and MusEd1, the system that was designed to test Presto1. Beaming was simple and deleting beams was not possible in MusEd1. None of the subjects were told about the logical or the visual rule. The subjects were allowed to experiment with the system for 15 minutes. Next, the subjects were given 15 minutes to complete two exercises in a questionnaire on paper. This questionnaire is reproduced in Appendix C.

The first exercise had twelve questions, and subjects were required to draw beam gestures over groups of notes that they thought should get the results stated in the exercise. The second exercise contained eight questions and subjects were required to draw the results, that is, draw beams over notes, that they would expect to get from the beam gestures that were in the questions. Subjects were allowed to draw more than one gesture or beam and to provide more than one answer in the exercises. Subjects were

also asked to briefly state their experience and training in music to help in the analysis of their answers.

When the subjects' answers were analysed, they were separated into three categories: the answers that used the visual rule, the answers that used the logical rule, and miscellaneous answers that used neither. The results of this analysis is decomposed by subjects and presented in Table 5.1. The total number of answers were different for each subject because all the subjects were allowed to give more than one answer to each question. Appendix D shows a set of model answers that uses the visual rule and another set of the model answers that uses the logical rule.

Subjects used the visual rule 43% to 79% of the time in their answers, with an average of 63.2%. They used the logical rule on an average of 12.2% of the time and gave miscellaneous answers on an average of 24.6% of the time. Twenty-two percent of the miscellaneous answers were errors because subjects misunderstood the instructions, and the rest of the miscellaneous answers were a modified version of the visual rule.

Subject	**Visual rule**	**Logical rule**	**Miscellaneous**	**Total**
1	16 (73%)	1 (4%)	5 (23%)	22
2	12 (60%)	8 (40%)	0 (0%)	20
3	14 (61%)	5 (22%)	4 (17%)	23
4	14 (70%)	0 (0%)	6 (30%)	20
5	12 (43%)	7 (25%)	9 (32%)	28
6	19 (79%)	0 (0%)	5 (21%)	24
7	16 (52%)	8 (26%)	7 (22%)	31
8	10 (45%)	0 (0%)	12 (55%)	22
9	15 (75%)	1 (5%)	4 (20%)	20
10	17 (74%)	0 (0%)	6 (26%)	23
Average (%)	(63.2%)	(12.2%)	(24.6%)	-

Table 5.1: Categories of subjects' answers in beam survey.

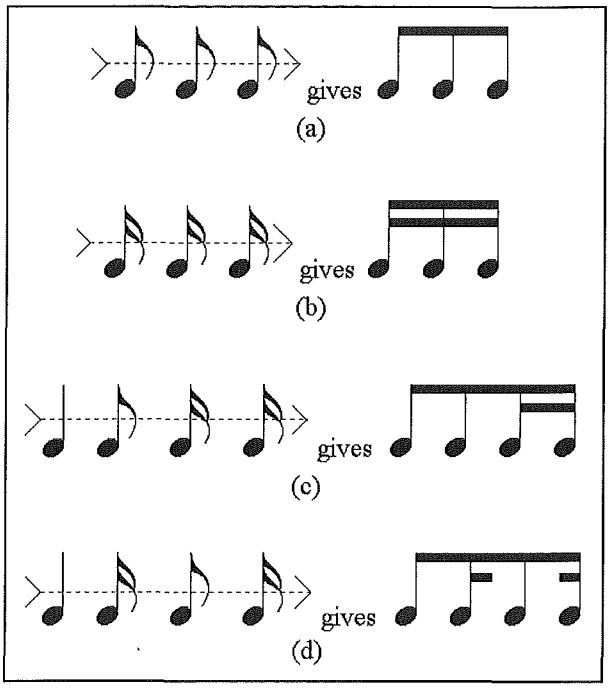


Figure 5.25: Add a new beam on individual notes using the visual rule.

The high percentage of answers that used the visual rule and the inclusion of a modified visual rule in the miscellaneous answers show that the subjects always used the visual rule more often than the logical rule in the survey and suggest that users prefer the visual rule when they draw the beam gesture.

5.5 Examples using the visual rule

This section demonstrates the beam gesture using the visual rule, which is also the paradigm of the beam gesture. Most examples in this section are from Section 5.2. Some examples are illustrated a second time using the logical rule, but deletion of beams, which is discussed in Section 5.6, is needed in these examples to get the same results using the visual rule.

5.5.1 Adding a new beam

In Section 5.2.1, two issues occur when the user adds a new beam. First, should individual quavers or notes with shorter durations be halved if the user draws a beam gesture over them? The beam gesture using the visual rule does not halve these notes because the visual rule affects only the gaps between the notes. Therefore, the quavers

will remain as quavers and have only one beam on them as in Figure 5.25a, and semiquavers will still be semiquavers with two beams on them as in Figure 5.25b.

Second, should the secondary beams of semiquavers or notes with shorter duration become complete beams or broken beams when these notes are beamed? The visual rule makes the secondary beams complete rather than broken, as shown in Figure 5.25b. This also applies to individual notes of mixed duration. Secondary beams will be connected into complete beams where possible, as illustrated in Figure 5.25c and Figure 5.25d. Figure 5.26 has more examples where the user adds a beam, and the visual rule is applied.

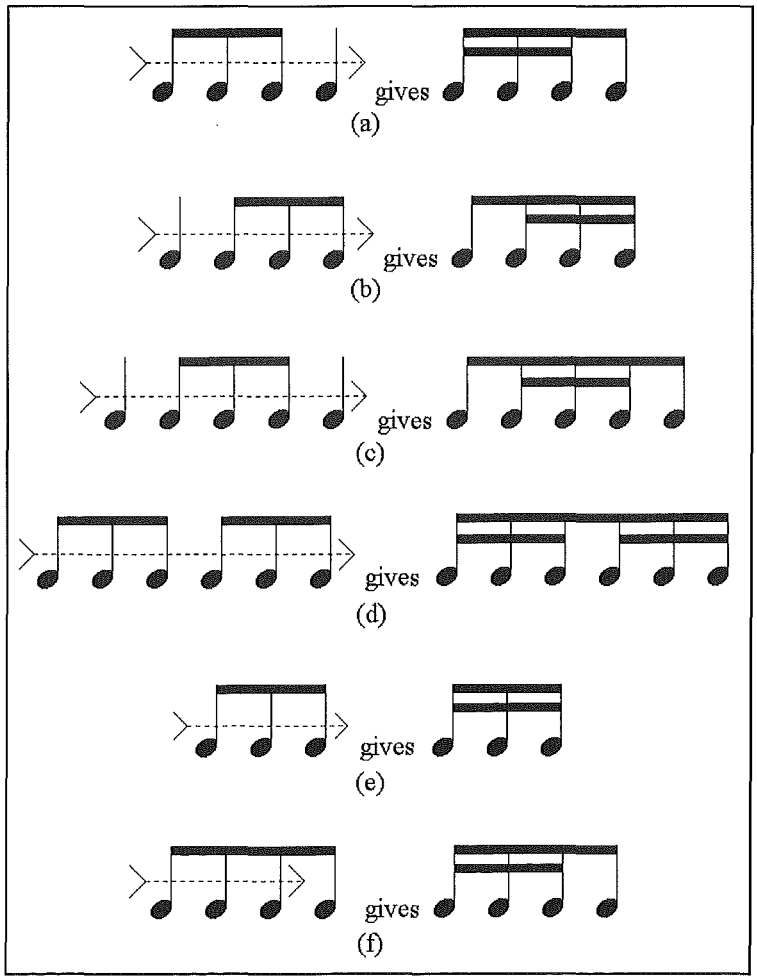


Figure 5.26: Add a beam on beamed groups using the visual rule.

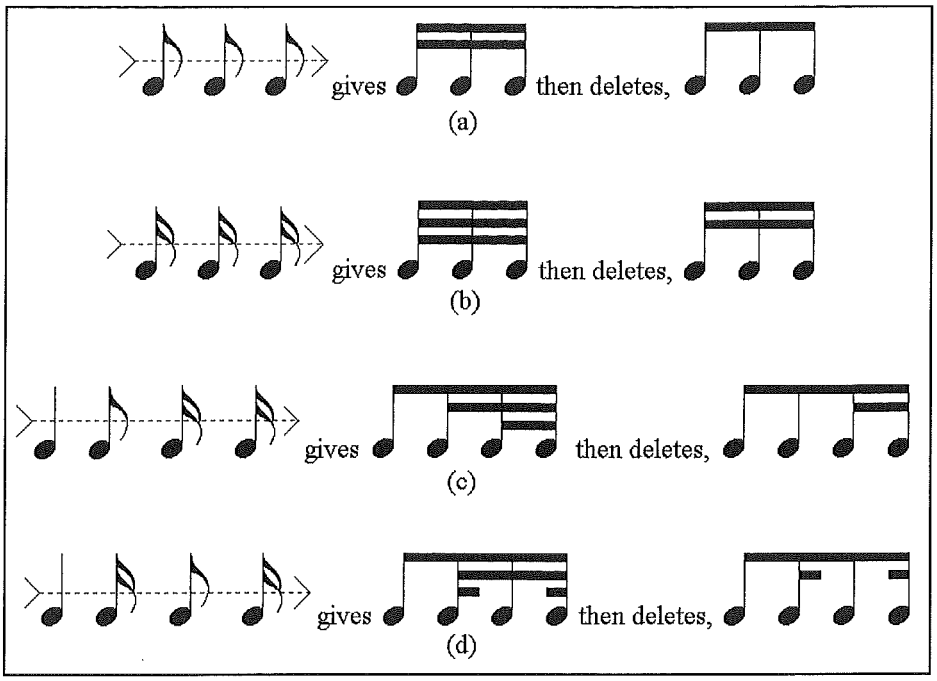


Figure 5.27: Add a beam using the logical rule.

Some of these examples are shown again in Figure 5.27, but they are drawn with the beam gesture that uses the logical rule. After the beams are drawn on the notes, some beams must be deleted to achieve the same desired results. The other examples are not repeated to demonstrate the use of the logical rule because their beam gestures are the same as those using the visual rule.

5.5.2 Connecting multiple beamed groups

There are three issues when a user connects multiple beamed groups in Section 5.2.2. Issue one is where the ends of a beam gesture should be in the beamed groups. The visual rule means that the left end of the beam gesture points at the last note of the first beamed group and the right end points at the first note of the second group. This is similar to Figure 5.16g, which is the shortest gesture among the nine possible combinations in Figure 5.16. This shorter stroke is preferred to longer ones, where Figure 5.16c is the longest, because it is the most efficient.

The second issue occurs if there are more than two beamed groups in the beam gesture. When a beam gesture is drawn across three beamed groups, the visual rule will not only connect the three groups, it will also add another beam in the gap between the two middle notes of the whole group, as shown in Figure 5.28a. If the user has drawn a beam gesture over two beamed groups that have individual notes between them, the visual rule will only connect the groups into one beamed group, as in Figure 5.28b.

The third issue occurs when the user connects two beamed groups and both groups have secondary beams. In this case, the visual rule connects only one beam as in Figure 5.28c. The user will need to draw extra beam gestures to connect the secondary beams between the two groups, as illustrated in Figure 5.28d.

The gestures for these examples are different if the beam gesture uses the logical rule, as illustrated in Figure 5.29. The user must delete some beams after beaming the notes to get the same desired effects.

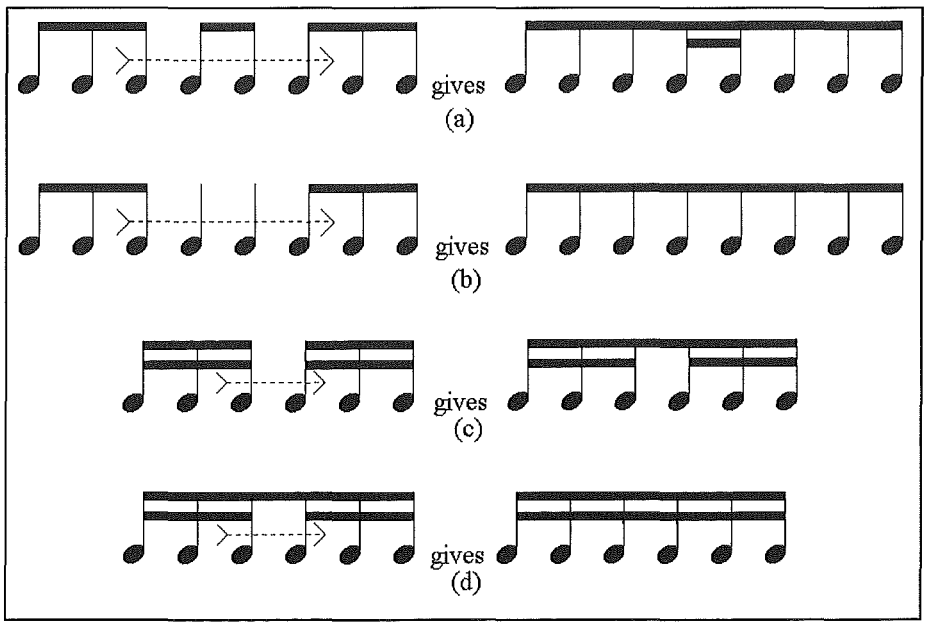


Figure 5.28: Connect multiple beamed groups using the visual rule.

5.5.3 Extending a beamed group

When the user extends a beamed group, three issues arise, as discussed in Section 5.2.3. First, there is the issue of where the end of the beam gesture that is in the beamed group should be. The beam gesture that uses the visual rule is like that in Figure 5.19c, that is this end should point at the note nearest to the notes that are not in the beamed group, shown in Figure 5.30a and Figure 5.30b. This is the shortest and most efficient gesture compared to the other two possibilities in Figure 5.19a and Figure 5.19b.

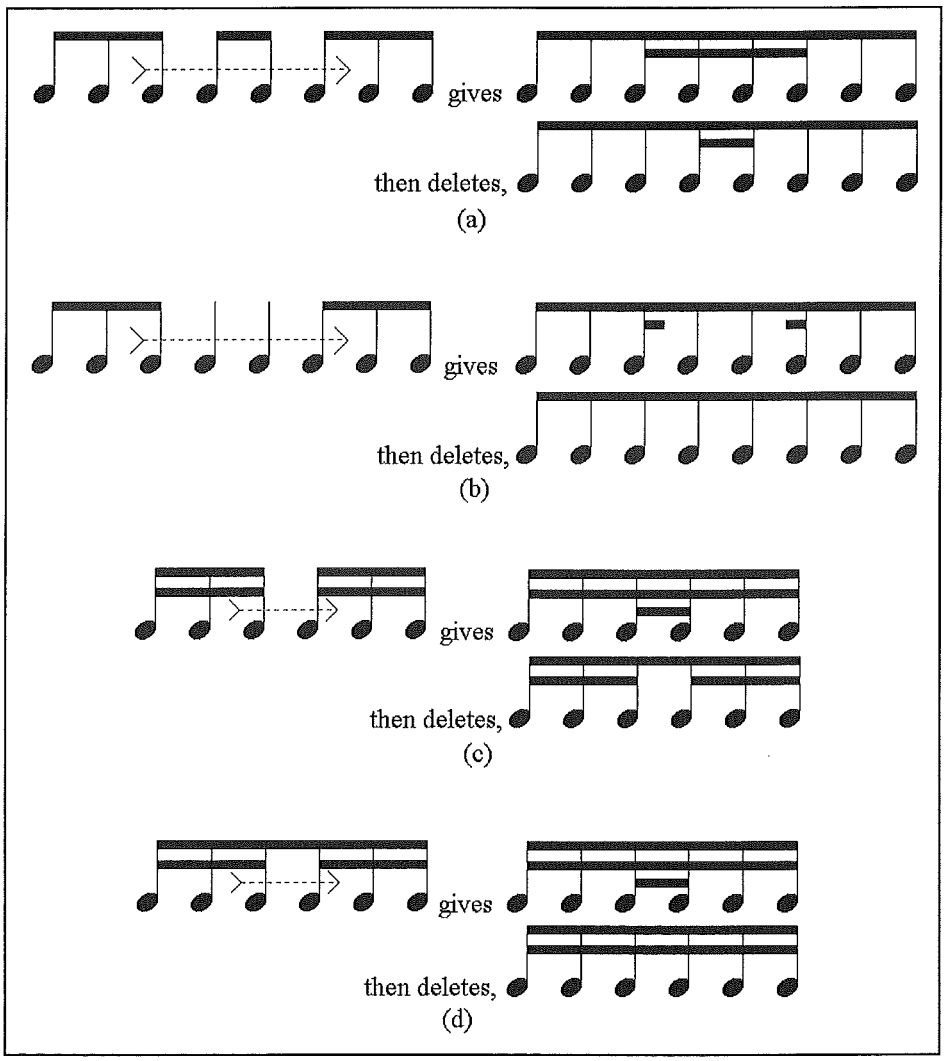


Figure 5.29: Connect multiple beamed groups using the logical rule.

The second issue arises when the user tries to connect both ends of a beamed group with one beam gesture. According to the visual rule, the user draws two beam gestures instead of one to connect two ends of the group, as in Figure 5.30c; one gesture is drawn on the left of the group, and the other is drawn on the right.

The third issue happens when the user extends a beamed group that has secondary beams. Similar to connecting two beamed groups, the visual rule extends only one beam of the group as in Figure 5.30d. Here, if the user wants to extend the secondary beams of the beamed group, the user has to draw extra beam gestures as in Figure 5.30e.

The beam gestures using the logical rule are different for these examples, as illustrated in Figure 5.31. In addition to the gestures, the user must also delete some beams to achieve the same desired results.

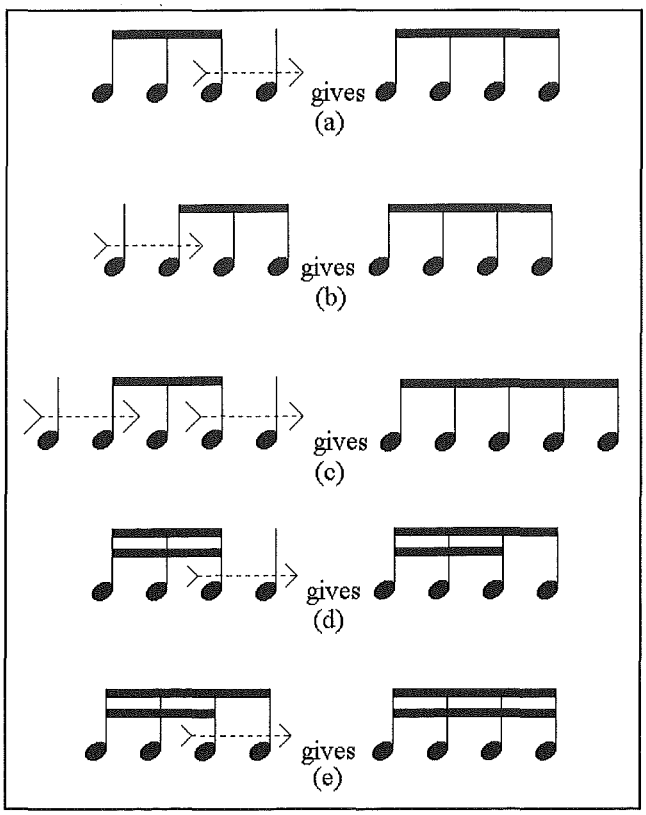


Figure 5.30: Extend a beamed group using the visual rule.

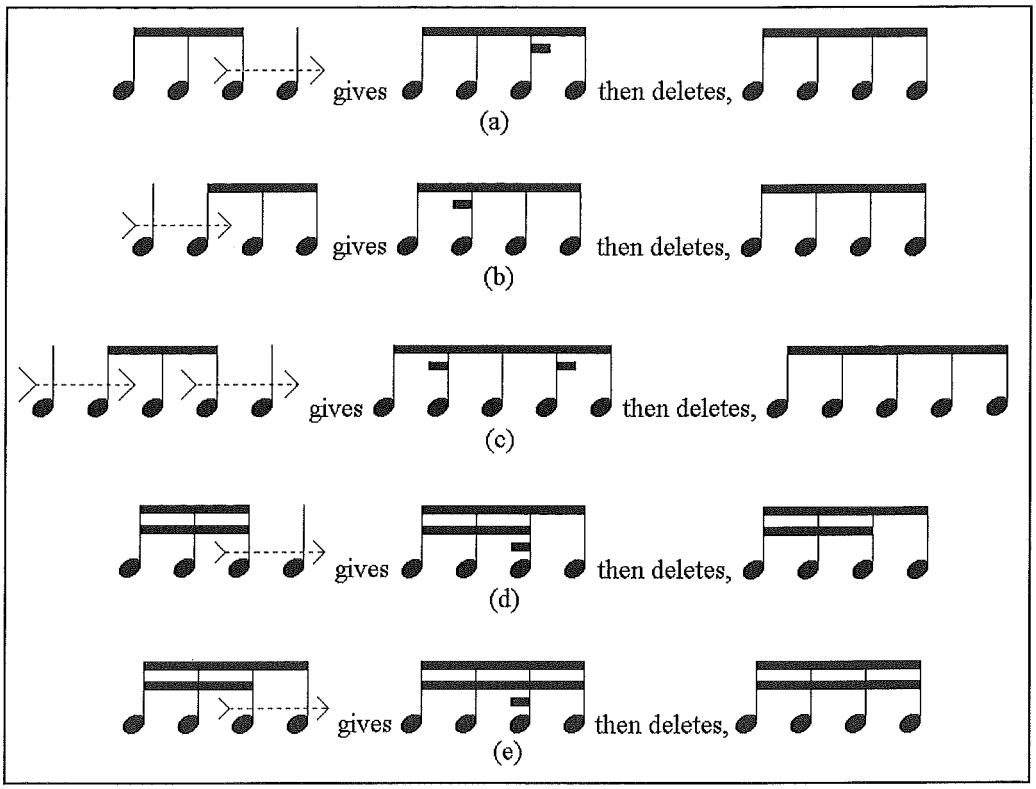


Figure 5.31: Extend a beamed group using the logical rule.

5.5.4 Adding and editing broken beams

The gesture for drawing a broken beam in Presto1 is the same as that for drawing a complete beam, except that it is shorter and drawn over only one note. There are two issues associated with drawing broken beams. The first issue is how the user should express the decision to point the broken beam to the left or to the right. Another issue is how the user should extend a broken beam or connect multiple broken beams, which is discussed in Section 5.2.4.

One possible solution to express the direction that the broken beam points to is to draw the stroke in that direction from the stem of the note. The stroke is drawn to the left if the broken beam points to the left as in Figure 5.32a, and vice versa as in Figure 5.32b.

This method has three disadvantages. First, the user has to remember the different directions of the same gesture for pointing to different sides. This can result in longer cognitive recall, and in turn slows music input. This pause is significant, because the delay between each gesture dominates input time [gold93]. Second, the beam gesture

itself does not differentiate the direction of the stroke, that is, there is no difference whether the gesture is drawn to the right or to the left. Since, the same beam gesture is used to draw the broken beam, only in a shorter version, the gesture should also allow for both directions of the gesture to take the same effect. Third, it may be awkward and unnatural for users who are right-handed to draw from right to left or for those who are left-handed to draw from left to right. An awkward gesture may cause a stroke to be crooked, and thus the gesture recognizer will treat it as an error or misinterpret it as another gesture.

Another solution is to draw the stroke, in any direction, on the side of the stem where the broken beam points to. To point a broken beam to the left, draw the gesture on the left side of the stem as in Figure 5.33a, and vice versa as in Figure 5.33b. If the user wants to change the side that a broken beam points to, the user can redraw it on the other side and the system will delete the original broken beam and draw a new one on the other side. The user also need to redraw only one broken beam gesture to change the side when there are more than one broken beam.

The second issue is whether the beam gesture extends a broken beam or connects multiple broken beams. In Section 5.3, the visual rule adds a beam in the gaps between the notes, but it does not add or delete broken beams. Thus, broken beams will be treated separately from complete beams; the user draws the beam gesture as illustrated in Figure 5.34, but it does not affect any broken beams. Deletion of broken beams is discussed in Section 5.6.

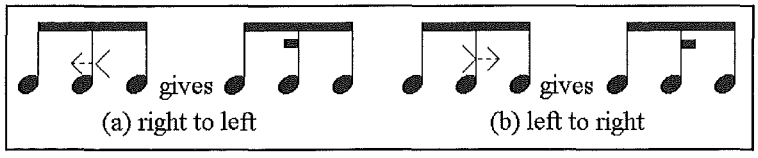


Figure 5.32: Gesture to place a broken beam.

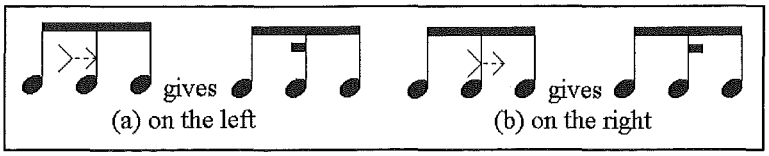


Figure 5.33: Another gesture to place a broken beam.

5.5.5 Adding and deleting notes in a beamed group

When the user adds a note in a beamed group, the user must draw a dot in the gap between two notes of that beamed group and the new note will adopt the number of beams in that gap, as shown in Figure 5.35. The black dot • in the figure represents the gesture that adds a note. If the user wants to add a note on the left or the right of the beamed group, the user has to draw the appropriate gesture to add an individual note and then draw the beam to connect that note to the beamed group.

Visually, when the user deletes a note from a beamed group, the half portions of the beams on each side of the note are also deleted as in Figure 5.36. Logically, the remaining notes in the beamed group will have the same duration as they were before the note was deleted.

5.6 Deletion of beams

Deleting beams is just as important as drawing beams. The importance of deletion and the design of its gesture in Presto2 is discussed in Section 4.1. The delete gesture in Presto2 is to press a button on the pen barrel or a key on the keyboard, and tap or draw a dot (button-and-tap gesture) on a musical symbol. MusEd1 does not allow users to delete beams.

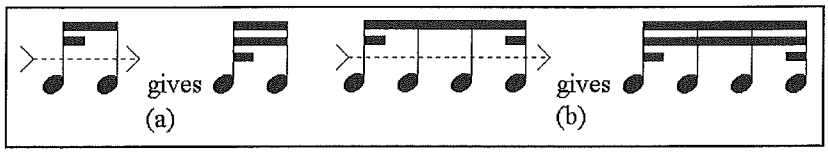


Figure 5.34: Gesture does not affect broken beams.

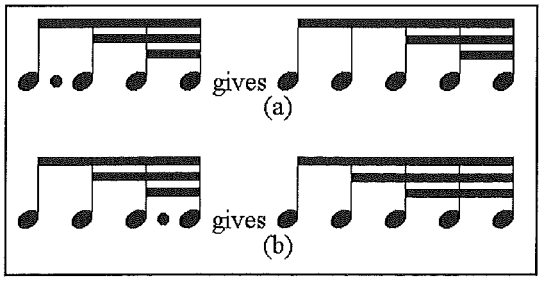


Figure 5.35: Add a note to a beamed group.

The user may need to delete a broken beam as shown in Figure 5.37a, a part of a beam in the gap between two notes as in Figure 5.37b, or a whole beam of a beamed group as in Figure 5.37c. The button-and-tap gesture is used to delete a broken beam, because a broken beam is an isolated musical symbol. To delete a broken beam, the user has to tap on the broken beam itself, as in Figure 5.38.

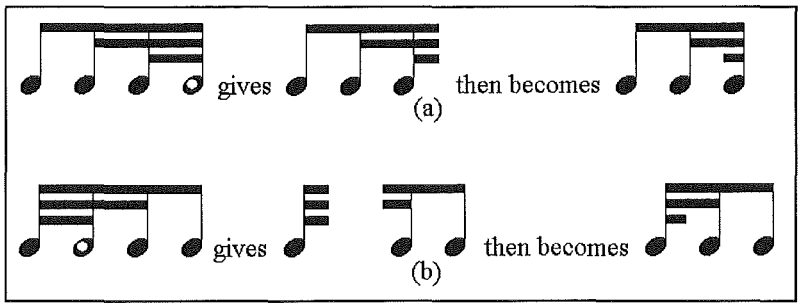


Figure 5.36: Delete a note from a beamed group.

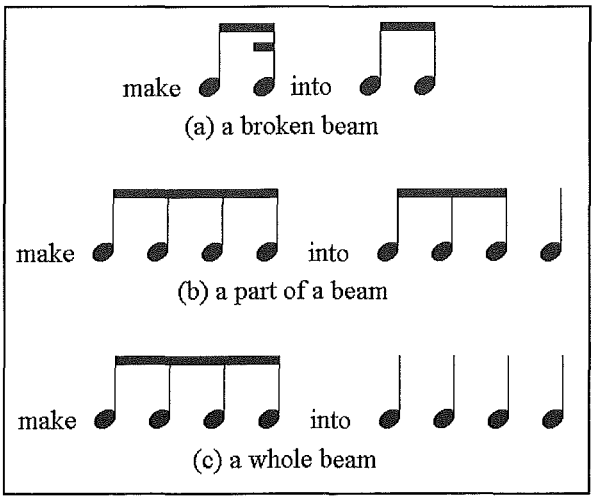


Figure 5.37: Delete a beam.

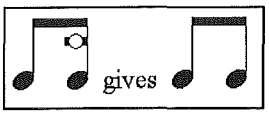


Figure 5.38: Delete a broken beam.

When the user deletes a part of a beam or a whole beam, the user can tap on the same position on the beam for both actions, so the system must know if the user wants to delete a part of a beam or a whole beam. A whole beam consists of many parts of the beam. A part of the beam is in the gap between two consecutive notes. One way to differentiate between deleting a part of a beam and a whole beam is to use a pop-up menu with these two options when the user taps on the beam. However, using a pop-up menu to assist in the execution of a common command such as deletion slows down music input. To solve the problem, two different gestures are needed to delete a beam partially and completely.

5.6.1 Deletion of complete beams

The user deletes a part of a beam when the user wants that part, but not any other parts of the beam, erased. The user can also delete a whole beam by deleting each part of that beam. For example, if the beamed group has three notes, the user deletes each part of the beam in the two gaps of the group. Similarly, if the group has six notes, the user deletes each part in five gaps. The user needs to delete the whole beam only if the user draws a gesture that is misrecognized as a beam gesture, or if the user draws the beam gesture unintentionally. The user can also undo the gesture. Deleting a whole beam is not needed as often as deleting a part of a beam. Therefore, the gesture for deleting a part of a beam should be faster than the gesture for deleting whole beams.

The user can use the delete gesture, which is the button-and-tap gesture in Presto2, to delete a part of a beam. The process of deleting a part of a primary beam is illustrated in Figure 5.39a, where the small white circle O represents the delete gesture. A primary beam is a beam on the first level. Figure 5.39b shows deleting part of a secondary beam. In this case, when the part of the beam is deleted, the beams of lower levels in the same gap are moved up one level, as illustrated in Figure 5.39b. However, the user sees only the first step and the final step in Figure 5.39b, and the user is actually deleting the beam on the lowest level in that gap.

The gesture for deleting a whole beam should be longer than one tap, which deletes a part of a beam, and shorter than or equal to two taps, which deletes each part of a whole beam of at least three notes. In addition, the gesture should be combined with the pen button to distinguish it as a deletion gesture.

One option is a line which goes up, down, left, or right. This is a gesture that can be drawn short to delete a whole beam. However, line gestures are already used to insert a

few musical symbols, which are listed in Appendix A, and the user may confuse this gesture with others. Another option is a double tap in quick succession. A double tap is longer than a single tap, and it is shorter than other gestures in Presto2. The user can remember it as an extension of the delete gesture to delete more than one part of a beam. Pressing the button and tapping on musical symbols are dedicated to deletion. The double tap may be competitive with the short line in speed.

The delete gesture deletes a part of a beam by touching only in the gap between two notes, whereas the specific gesture that deletes a whole beam can touch anywhere on the beam. Figure 5.40a illustrates the deletion of a whole beam. The white circle with a small black dot inside ⊙ represents the gesture for deleting a whole beam.

When there is more than one beam in a group of notes, the delete gesture will delete the beam according to the position of the gesture. For example in Figure 5.40b, the user deletes the first secondary beam, and the system will erase only that beam. The user can also use any of the two delete gestures to delete a beam in a group with only two notes.

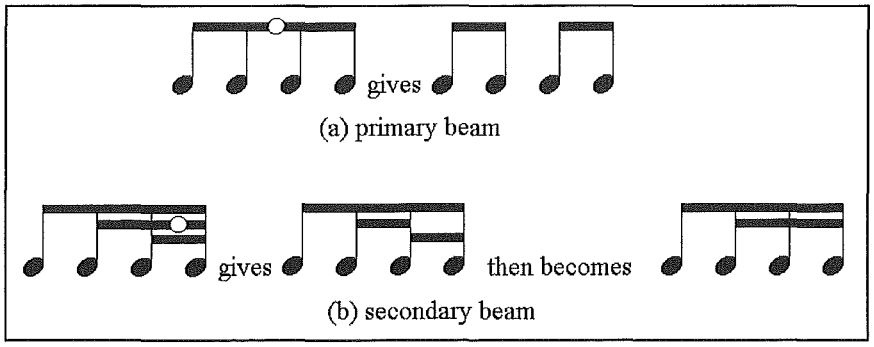


Figure 5.39: Delete a part of a beam.

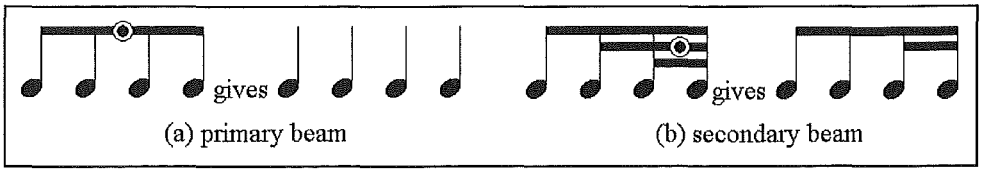


Figure 5.40: Delete a whole beam.

5.7 Tolerances of gestures

When drawing beam gestures, some users naturally draw carefully from stem to stem. However, fast input requires a quick stroke, in which case the start end and the finish end of the beam gesture will be inaccurate. Allowances for accepting both ways of input should be made for drawing and deleting beams.

For example, when a beam gesture is drawn from left to right, the start end can be close to the stem as shown in Figure 5.41a, touch the stem as in Figure 5.41b, or cross the stem of a note as in Figure 5.41c. However, if the start end stops close to or crosses the stem, how far should that end be from the stem? In other words, what is the *tolerance* to recognize that note and include it in the beamed group? The finish end of that gesture will have the same issue. Here, the beam gesture is drawn from left to right. If the gesture is drawn from right to left, the reverse situation will apply.

The area between the two solid lines in Figure 5.42a is the tolerance of the start end of the beam gesture, which is drawn from left to right. Thus, the start end should be between these two lines to include the second note in the beaming. The first and third notes in the figure are only representatives and can be replaced by other musical symbols. If p is the proportion of the space between the two notes that the start end is permitted to overrun and cross the stem of the second note, that is, between the left solid line and the stem, then $1\text{-}p$ is the proportion of that space that the start end is permitted to underrun and stop close to the stem, that is, between the stem and the right solid line. Figure 5.42b shows $p = 1$, where the start end of the beam gesture must overrun and cross the stem of the second note to include that note in the beaming. On the other hand, if $p = 0$, all the start end must underrun and not touch the stem of the second note in Figure 5.42c to include the note in the beaming. The value of p can also be between zero and one, and the start end is allowed to stop close to or cross the stem of the second note to a certain length, as illustrated in Figure 5.42a.

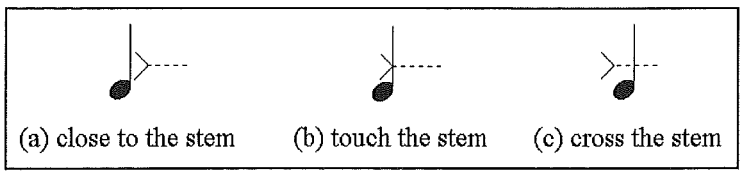


Figure 5.41: Start end of the beam gesture drawn from left to right.

The tolerance of the finish end of the beam gesture, which is also drawn from left to right, is the area between the two solid lines in Figure 5.43a. The previous ranges of the start end also applies to the finish end, but in reverse. Here, q of the finish end is equivalent to p of the start end, and $1\text{-}q$ is equivalent to $1\text{-}p$, as in Figure 5.43. The ranges of the start and finish ends will be reversed for beam gestures that are drawn from right to left.

Samples in the survey on the beam gesture in Section 5.4 were used to observe where subjects put the start and finish ends of the beam gesture. The start end of the beam gesture touched or crossed the stem of the note in 92.5% of the samples; only in 7.5% of the answers that the start end was close to but not touching the stem. Similarly, the finish end of the beam gesture touched or crossed the stem of the note in 94.2% of the samples in the survey, and only in 5.8% of the cases that the finish end was close to but not touching the stem. Therefore, the tolerances of the start and finish ends should allow for more overruns and less underruns, that is, the values of p and q should be more than the value of $1\text{-}p$ and $1\text{-}q$.

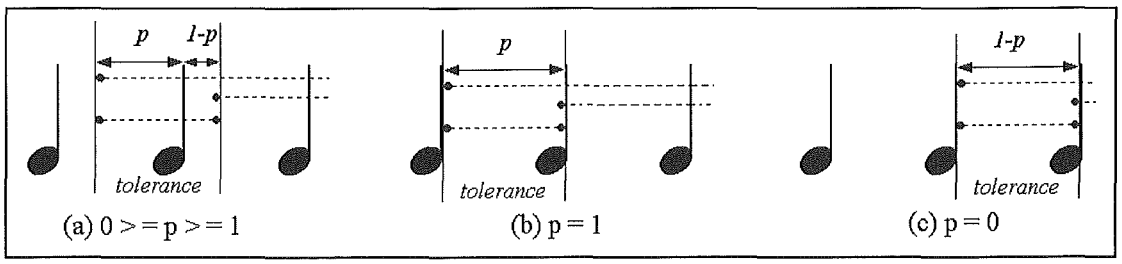


Figure 5.42: *Tolerance* of the left end of the beam gesture.

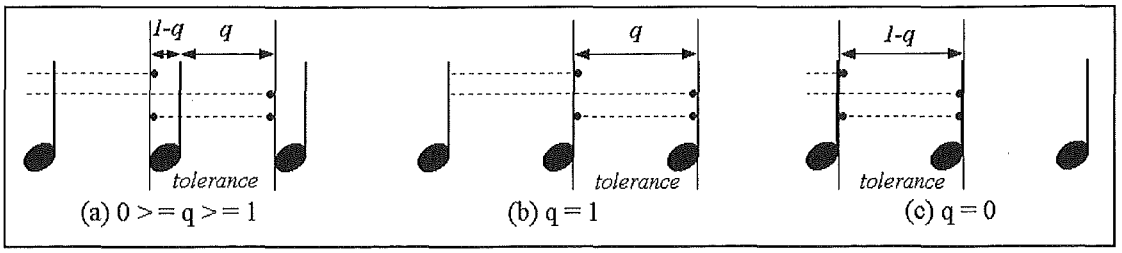


Figure 5.43: *Tolerance* of the right end of the beam gesture.

The tolerance for drawing a broken beam that points to the left should have more area on the left side of the stem and the tolerance for drawing a broken beam that points to the right should have more area on the right side. The tolerances for drawing the complete beam can be used to define the tolerances for drawing the broken beams.

The tolerance for drawing a left broken beam, that is, the third dotted line, is within the two solid lines in Figure 5.42a; p is the proportion of the broken beam where the left end of the gesture should be, and $1\text{-}p$ is the proportion of the broken beam where the right end should be. The tolerance of the right broken beam gesture is within the two solid lines in Figure 5.43a. The proportion of the broken beam where the left end of the gesture, that is the third dotted line, should be is q, and the proportion of the broken beam where the right end should be is $1\text{-}q$. The broken beam gesture has a minimum length, thus most part of the left broken beam gesture must be drawn on the left side of the stem and most part of the right broken beam gesture must be drawn on the right side.

The tolerance for deleting beams should also be defined. How far should the deletion gesture be away from the beam? If the dot is within the beams, its vertical position does not matter, except for deleting a whole beam where the system needs to know the specific beam to delete. One solution is limit the gesture to be within the boundaries which is half of the space between two beams, as shown in Figure 5.44.

5.8 Data structures for beams

This section looks at four different data structures for storing information on beams. They are the *parenthesis* structure, the *tree* structure, the *visual* structure, and the *lime* structure. Figure 5.45 shows an example beam group, which will be used to illustrate the four data structures for beams.

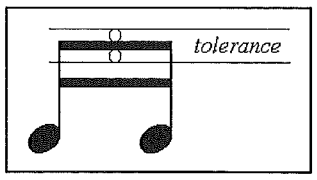


Figure 5.44: *Tolerance* of the delete gesture on a beam.

5.8.1 Parenthesis structure

Musical symbols in MusEd1 are stored in a linked list with the parenthesis structure to store information about beams. This structure treats a beam like a pair of parentheses; the start of a beam is like an opening parenthesis and the end of a beam is like a closing parenthesis. The structure adds a beam-on node and a beam-off node in between note nodes wherever there is a beam. Figure 5.46 illustrates the parenthesis structure of the beamed group in Figure 5.45. Extra information on the beams are stored as attributes of the beam-on node. The beam-on node will also have a pointer to its corresponding beam-off node, and vice versa.

The strength of this structure is that it clearly marks where each beam starts and ends. The beams are explicit in the structure and need not be inferred. This structure has some weaknesses too. First, it does not work well with the visual rule. Its operations for editing and deleting the beams are complicated. It does not represent each gap between the notes like the visual rule, instead it represents the beamed group as a whole. Also, the pairs of beam-on and beam-off nodes have to be checked to make sure they pair properly, because it is possible to store syntatically incorrect structures.

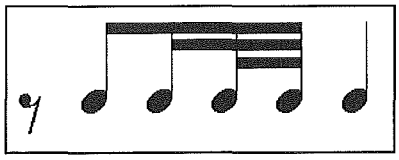


Figure 5.45: Example of a beamed group.

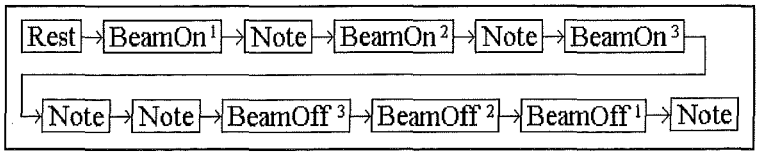


Figure 5.46: Parenthesis structure of the beamed group.

5.8.2 Tree structure

Another way of storing beams is to use a tree structure. Buxton et al. [buxt78] and Dannenberg [dann93] recognize that beams should be in a hierarchy structure. Bainbridge [bain96] also points out that if a group of notes with beams is rotated 90 degrees, the group looks like a programming language with nesting in it. Nesting is a form of hierarchy and it is illustrated in Figure 5.47.

The tree structure has only one beam node to represent a beam, and information about the beam is stored in this element, as shown in Figure 5.48. This beam node also contains attributes such as a linked list of note nodes and other secondary beam nodes that belong to this beam group.

5.8.3 Visual structure

The visual structure has a beam node in the gap between the note nodes where the beams are, as in Figure 5.49, and the information about the beam, particularly the number of beams, is stored as attributes in the beam node. A visual structure suits the visual rule very well. The strength is that it is directly related to the visual representation of the beams.

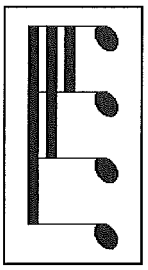


Figure 5.47: Side view of the beamed group.

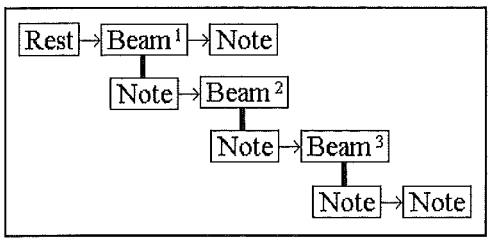


Figure 5.48: Tree structure of the beamed group.

5.8.4 Tilia structure

This structure is written in Tilia music representation and used in the Lime music editor [hake93], which is described in Section 2.2.4. There are no explicit beam nodes and all the information on beams are stored as attributes in the note nodes. The overall structure is in Figure 5.50. This structure has a manipulative advantage over the other three structures that are reviewed in this section; without an explicit beam node, the lime structure may be easier to do insertions, deletions and presentations on the screen.

5.9 Implementation

The beam gesture in Presto1 is retained in Presto2 and implemented in MusEd2. MusEd2 interpretes the beam gesture according to the visual rule, which is the paradigm discussed in Section 5.3, to capture all the possible scenarios. The ends of the beam gesture have considerable tolerances; the values of p in Figure 5.42 and q in Figure 5.43 both take two-thirds. However, if the beam gesture includes the last note of the score and there are no symbols on the right side of that note, that end of the beam gesture can be an arbitrary run across the score. The algorithm for calculating the correct positioning and slope of a beam in Presto is from Assayang and Timis [assa86]. An example of beamed groups in MusEd2 is in Figure 5.51.

Deletion is also implemented in MusEd2. The beam structure used in MusEd2 is the Tilia structure, because it has fewer nodes to manipulate in edit operations than the other structures in Section 5.8. This advantage eases the task of capturing various beaming scenarios and the task of programming the process. In addition, this structure can also be expanded to include exceptional beamings like those in Figure 5.4. Other structures described in Section 5.8 may be more difficult to implement but can also be used.

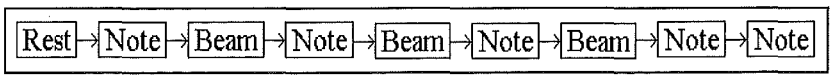


Figure 5.49: Visual structure of the beamed group.

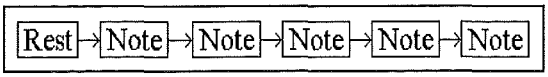


Figure 5.50: Lime structure of the beamed group.

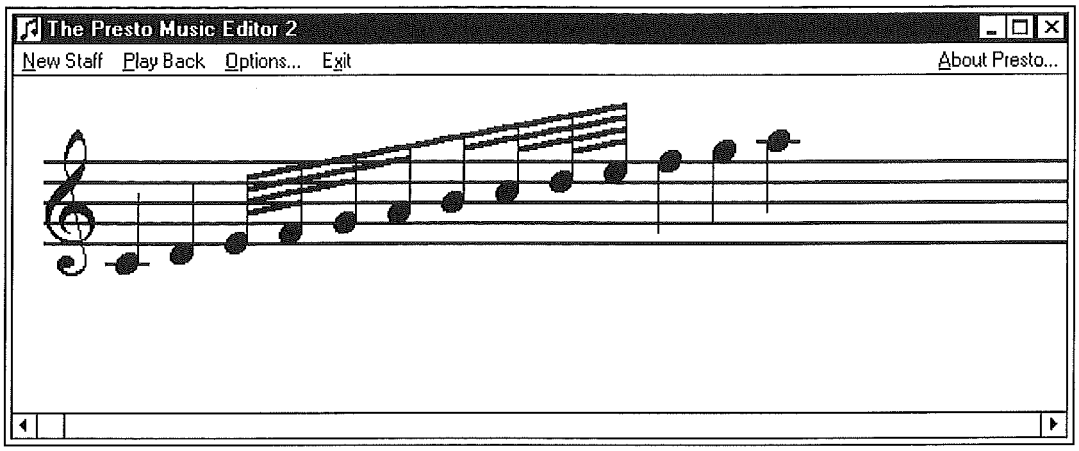


Figure 5.51: Beamed groups in MusEd2.

5.10 Summary

This chapter has refined the beam gesture in Presto. A paradigm for the gesture is formed according to the beam rules in CMN and issues about beaming in Presto. This paradigm uses the visual rule to interpret the beam gesture. The user survey has confirmed that the visual rule is suitable for musicians to draw beams as well as removed complicated sequences of gestures sometimes required by the logical rule. This chapter has also discussed the deletion of beams, the tolerances of the beam gesture, and appropriate data structures to store beams in MusEd2. Finally, the implementation of beams in MusEd2 is presented.

Chapter 6

Feedback

Feedback is an important part of human-computer interaction [barf93, pere96]. Norman [norm$_1$88] defines the feedback of a system as "sending back to the user information about what action has actually been done, what result has been accomplished." For example, there is no feedback if you cannot hear yourself when you are talking or if you cannot see the marks made when you are writing. The absence of feedback will cause the performance to be slower and harder than if there is feedback. In a two-way communication, feedback allows for flexibility and correction. Feedback is also a well-known concept in information theory [norm$_1$88]. For instance, a person talking to another person can alter or cut short a sentence if there is feedback of understanding from the other person.

The information sent in feedback from a system can include results due to the user's action, or the state of the system which is not affected by the user's action [pere96]. If the feedback from a system is the immediate effect of the user's action, the feedback tells the user what was done and helps the user to perform better in the system. If the feedback is the result of the current state of processing in the system, the user will be better informed and not feel frustrated and ignorant about what the system is doing. In this way, the user can learn more effectively and build up a good user model of the system, and the designers can build a more robust system that prevents errors [barf93, norm$_1$88].

On the other hand, incorrect feedback can create problems in the interaction process between the user and the system. Incorrect feedback includes [barf93]:

- absent feedback, for example there is no indicator that a lift button is pressed or not;

- feedback that is badly designed, such as lighting up an indicator when the lift arrives but the indicator does not show whether the lift is going up or down; and
- feedback that is distorted by errors or obstructed by other parts of the system, for example if the arrow, that indicates the direction a lift is going, is faulty and does not light up when the lift arrives, or if the meaning of the arrows are swapped.

A system can send feedback to the user by means of the five senses: hearing, sight, smell, taste, and touch [barf93]. The senses of taste and smell are difficult to implement into a system and are not practical to use for prompt reaction to changes, thus they are rarely used. The more frequent feedback channels are visual (sight), audio (hearing), and tactile (touch). A combination of these channels are often used. For example, animated icons (visual) can be enhanced with sound effects (audio) [baec95$_c$].

This chapter first reviews the benefits and drawbacks of visual and audio feedback. Then it looks at some issues of feedback in Presto, which is the research prototype that consists of the gesture set and the music editor. Tactile feedback involves hardware issues and is beyond the scope of this research, thus it will not be discussed in this thesis, although the system's key attraction is that the tactile experience which is similar to paper.

6.1 Visual feedback

Visual feedback is the most common form of feedback found in systems [barf93, brew94$_b$, baec95$_b$]. Harrison [harr95] conducted an experiment to compare still, animated, textual, and spoken on-line help. She concluded that although the users who received spoken help could do more tasks faster and more accurately than those who were given visual feedback, many of them preferred visual over audio feedback.

Visual feedback can be provided in the form of still images or animation [jack97]. Visual feedback that uses still images shows the user only one or two images. For instance, before a *rotate* command is issued, the system displays the object in its original position on the screen. After the rotate operation, the system shows the object in the rotated position [norm$_2$86]. On the other hand, animation in interfaces is defined as "a series of varying images presented dynamically according to user actions." [gonz96] In other words, animation uses a sequence of frames to show the gradual changes in the information on the screen. In the same example, an animated feedback of the rotate

operation will display the smooth and gradual movement of the object from its original position to the rotated position in a series of frames. Animation can demonstrate changes that still pictures cannot, because the user can track the animation visually and understand the changes easier than by viewing only still images; movement in the animation helps the user interpret what has happened and where the objects have gone [norm$_2$86].

Another example of animation is when the user executes an *undo* command, the system reverses and redisplays all the actions performed between the previous version and the current version of the system, without the pauses that the user might have between the actions [thim90]. Myers [myer85] has also used animated icons to provide information about the status of the system; this is the *progress indicator* which is now common in many systems. These examples show that animation is useful and helpful to users in their interaction with the system.

Despite the benefits, visual feedback has its drawbacks too. Since most of the information is presented through the visual channel, users can miss out information because their visual channels are overloaded or they are not looking in the right place at the right time.

6.2 Audio feedback

Although audio feedback (sound) cannot communicate as much information and is not as widely used as visual feedback [barf93], it is still useful and can provide information that cannot be made available in other ways [blys82, gave93, myna94$_a$, myna94$_b$]. Every day, we rely on sound to confirm our actions. Sound can be directly designed and included in a system, or it can already be part of the way the system works. For example, an alarm is designed to ring when there is a fire, but a conventional door would bang automatically when it is shut. In technology, sound is mainly used to give feedback that an event is happening or has happened.

Sound is becoming popular in user interfaces because it has been proved to have potential benefits in many human-computer interaction studies:

- Strict attention is not needed to detect sound since sound can be heard from 360 degrees [brew93, brew94$_b$]. It provides greater flexibility than visual feedback by not distracting the user's attention and alerting the user to changes or mistakes

effectively, especially if the user is away from the system or not looking at the monitor. This is important when the user needs to focus on more important tasks instead of the system while relying on the subconscience to detect significant sounds from the system [buxt85, kram94]. Sound can also attract the user's attention in a complex system in which the user cannot notice all visual feedback [gave91].

- Sound can convey information that is difficult to present through other channels, such as messages in windows that are occluded or if the screen is limited in size [brew94$_b$, gave95]. The system can also use sound to convey information that is not available in other methods; for instance, information about the current mode of the system or the current status of a process especially when the process is finished [brew93, mere97].
- Sound complements the visual display effectively and prevents the visual channel from being overloaded [brew94$_a$, brew94$_b$]. It can increase the amount of information communicated to the user and at the same time reduce the amount that the user has to receive visually [brew93]. For example, if the screen is already cluttered, audio cues will limit or decrease the number of messages on the screen [buxt85].
- Sound is immediate [gave95], and it can communicate information at a rate which can keep up with the pace of interaction [brew95].
- Sound that is well designed and used in systems may be more appealing and easier to learn than other forms of communication [buxt85, kram94].
- Human response to sound is quicker than response to images [kram94].
- Sound can help visually impaired users to access computers more easily [buxt85, itoh90, brew93,mere97].

Although sound benefits human-computer interaction, it must not be overused because it has its drawbacks too. Sound is not effective and unreliable when:

- it is annoying and distracting to the user as well as other people in the area [gave95, gali96];
- it is intrusive and difficult to keep private, unless the volume is low or the user wears earphones although earphones are not always desirable, may be uncomfortable, or may interfere with other tasks [brew94$_b$, kram94];
- it interferes with human-to-human communication at work [kram94];
- it is overused, then it will be ignored [gali96];

- users turn down the volume, turn the sound off, or do not hear it at all [kram94, gali96];
- it is not persistent enough to be detected later, it does not produce hard copy for distribution, and there is no record of its occurrence [kram94]; and
- users have limitations in hearing, such as being hard of hearing or tone-deaf [kram94].

Due to the drawbacks of sound, it must be used carefully. There are a few rules that can be followed when designing sounds in systems:

- allow the user to adjust the volume or turn the sound off [gave95];
- sounds should be concise and unobtrusive [gave95];
- there should be a mapping between sounds and the information that they convey, for instance, opening a directory sounds like opening a drawer [norm$_1$88, gave95];
- use sounds that fit in with the environment, for example, the length of a sound for alerting the user should be short instead of long [gave95];
- use six different sounds or fewer, so that users can discriminate among them [gave95, gali96];
- use unique but similar sounds consistently for similar situations, for example, systems use similar beeps for different system errors [gali96];
- the length of each sound should last at least 0.125 seconds [brew94$_a$]; and
- the delay between sounds should be at least 0.1 seconds [brew94$_a$].

6.3 Feedback in Presto

It is natural to combine visual and auditory feedback in the interface [brew94$_b$, gali96]. Research shows that feedback that consists of sound and graphics is more effective than using graphics alone [brew95, obor95]. Psychological evidence suggests that sharing information across different sensory channels increases redundancy in information and gives the user more than one chance to notice the information. As a result, combined feedback improves users' task performance [brew93, gave94], decreases learning times, reduces fatigue, and increases enthusiasm [kram94]. Thus, visual and audio feedback may be used to help users perform better in Presto.

This section designs several feedback features in Presto. First, the section looks at the issues of feedback when the user draws notes that are not on the staff, and inserts or deletes musical symbols. Other issues that are discussed include requiring the system to play back symbols, to show help in alternative gestures of the same symbol, and to send error messages. Since sound can also be irritating, the last part of this section suggests options to switch on and off the various audio feedback.

6.3.1 Notes not on the staff

Users have difficulty aiming at the desired pitch in MusEd1, the system developed to test the Presto1 gesture set that Anstice [anst96$_a$, anst96$_b$] designed, when they are entering notes that are above or below the staff, because there are no ledger lines above or below the staff. In MusEd1, the user guesses the position to enter a note above or below the staff. If the system does not insert the note on the desired pitch, the user has to use the appropriate gestures to raise or lower the note until it reaches the desired position.

One solution to this problem is to use a guide or scaffolding like the *mosaic* in Gesture Mosaic which is described in Section 2.1.3, or the *ladder* in Char-rec described in Section 2.1.5. A ladder of ledger lines can guide the user to aim for the pitch above or below the staff and draw the desired note. The guide can appear whenever the tablet senses the pen above or below the staff.

6.3.2 Insertion and deletion

Musical symbols, including notes and rests, can be inserted and deleted in the score with gestures in Presto. Sometimes, users insert and delete symbols without receiving clear confirmation that the system has inserted or deleted the correct ones. In MusEd1, a symbol appears when the user inserts that symbol and a symbol disappears when the user deletes it. The manuscript is also rearranged. Sound is used only when the user enters a note; the system plays back the pitch of the entered note, but the duration of the note is disregarded. There is no additional feedback on the insertion and deletion of symbols in MusEd1.

There needs to be visual feedback when symbol is inserted or deleted, because unlike text editors, the music editor does not have a caret to help position the insertion or deletion of symbols. With visual feedback, the user can easily know that the correct

symbol is deleted or inserted in the correct place, even if the user has lifted the pen and the cursor has moved elsewhere on the screen. In addition, audio feedback helps the user to know quickly that they have used the correct gesture and obtained the desired result.

A symbol that is inserted could be animated to appear gradually from light to dark, that is, from being blur to being clear. A simpler solution is to initially display the symbol differently, with another colour or additional markings, from the standard (black) symbols when it is inserted and to persist long enough for the user to notice. Then the symbol changes back to its standard appearance. Any symbol that the user chooses to delete can also be animated to change its image from dark to light and disappear eventually. This solution can be simplified to only changing the symbol to a different appearance, as it is for insertion, before it disappears from the score.

When the user inserts a note, the system should still play the pitch of the note but the sound can also reflect its duration. Rests do not have a pitch, so the system can play a unique sound for inserting rests and can last as long as their duration. However, playing longer notes or rests, such as the semibreve and breve, may have undesirable side effects; this issue is discussed in Section 6.3.6. The system need not produce sounds for symbols that are not notes or rests, because these symbols are not as common as notes or rests. Producing sounds for them will be unnecessary and will clutter the user's audio space. The frequency of musical symbols is in Anstice [anst96$_a$].

When the user deletes a symbol, the system can also produce a *pop* sound to alert the user that the symbol is deleted. When implementing the sound, it should be considered whether the sound is produced when the symbol changes its appearance or when it disappears; the former may be better. First, playing the sound when the symbol changes its appearance is consistent with insertion which also plays when the symbol changes to a non-standard appearance. Second, the user will still see the played symbol; it is the symbol that is non-standard.

6.3.3 Playback

Playback is a common feature in music systems; some of these systems are presented in Section 2.2. It enables musicians, especially composers, to *proof-listen* to what they have written on the manuscript. This valuable feature is not available in MusEd1; a note is played only when it is inserted or its pitch is changed.

When a system plays back the symbols of the manuscript, the user will hear only the pitch and the duration of the notes consecutively, interweaved by the silence according to the duration of the rests. As the score is played, the current musical symbol can be highlighted to guide the user along the score and the manuscript will be scrolled along too. Animation of playback by tracking each symbol is preferred to other methods, such as tracking only each bar or no tracking at all [pick97]. However, users may prefer to have different options of tracking.

Besides being able to listen to the whole score, the user should also be able to listen to a group of notes or rests when they are edited. The system can play an individual note when its pitch changes, when its duration changes, or when a beam is added to it. The user will then be able to listen to changes that is made to that note. When the user changes the duration of a rest, the system can play a sound for it too. This sound will be similar to that when it is inserted, as suggested in Section 6.3.2. Thus, the user will be able to listen to its duration. Finally, the system can play back the group of notes and rests when they are selected and edited; these edits include changing the pitch and the duration of the selection. The effect of playing notes or rests with long duration and a group of notes is discussed in Section 6.3.6.

6.3.4 Help in gestures

When the user is experimenting with Presto, it will be useful if the system can suggest alternative ways, if there are any, to obtain the same symbol or effect that the user has just achieved. For example, the system can suggest the other gestures for changing the duration of a note or a rest in Presto if the user has drawn one, because these effects have two sets of gestures, as described in Section 3.4.2. This on-line help can also be extended into a help system that requires an extensive knowledge base of gestures. This intelligent help system can be used to suggest shorter combinations of gestures to achieve effects. Figure 6.1a shows an example where the user halves two crotchets one by one, then draws a beam over them. In this case, the help system can suggest that the user directly draws a beam over the crotchets like that in Figure 6.1b. In these situations, the system needs to know the possible longer ways of obtaining results and their relevant shortcuts.

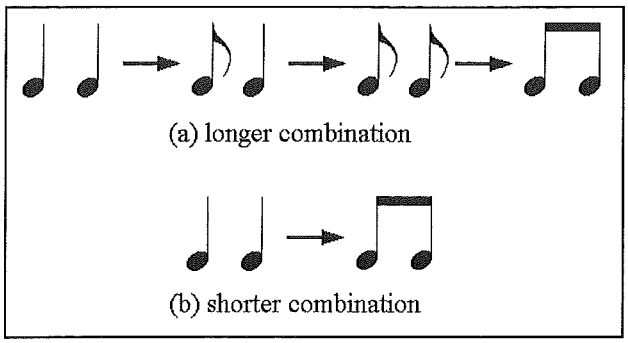


Figure 6.1: Alternative combinations of gestures.

One solution is to use a dialog box and a *beep* sound to inform and explain about the alternative gesture, but it will be too explicit and irritating to experts. Users may also have the false impression that there is an error in the system because this is the common purpose of the dialog box and the beep. The purpose here is not to warn the user but only to inform. If a dialog box is used to inform, it should have an option to disallow it from appearing again.

Another solution is to put the information in the status bar or at the bottom of the window. In addition, a soft sound, rather than a beep, can be used to bring the user's attention to the suggestion and have a subtle effect at the same time. Alternatively, the system need not use sound but could flash the information in the message for a few times instead. In addition, the system can give the user option to turn the messages off; this is discussed in Section 6.4.6.

6.3.5 Errors

There are two kinds of error warnings in Presto: system error warnings and gesture error warnings. Examples of system errors are when the system is unable to load the recognizer and when one or more files of a system are missing. Warnings about these errors are important and the user should not ignore or be unaware of it, because the errors indicate that the system is not executing properly or not executing at all. Presto uses a dialog box and a beep sound to warn users about system errors.

There are three situations when gesture errors will occur:

- a gesture is not recognized by the recognizer if that gesture is not in Presto;

- a gesture is recognized but it is drawn on an incorrect context, for instance, a gesture to raise notes is drawn on a rest which has a fixed position and cannot be raised; and
- a gesture is recognized and the context is correct but the gesture is not executed due to delimitations that are set, for example, the user draws a gesture that doubles the duration on a breve but the breve is the note with the longest duration in the system.

These warnings are not as crucial as system error warnings and they will not affect the system if they are ignored or overlooked. Moreover, users can make a lot of mistakes in the gestures when they are experimenting with the system. If warnings about these mistakes require the same amount of attention as the system error warnings, users may find it tedious to acknowledge them every time. Instead, users will prefer to try the gestures again without much hassle. Thus, these warnings can be informed in the status bar or at the bottom of the window, and flashing the warning or playing a soft sound can alert the user to the warning.

6.3.6 Options

The length of feedback is important, especially the length of animation and sound. If the length is as short as a beep, it is tolerable. However, if the length is longer, for example, playing a long note or a selection of notes, the system may either force the user to wait until the feedback has finished or allow the user to interrupt the feedback. If the system forces the user to wait, the system will not accept any commands from the user until after the feedback has ended. This solution may force and help a novice to learn more about the system, but an expert may find it tedious and may want to continue working in the system instead of waiting for the feedback to finish.

The second solution allows the user to control the length of the feedback. In Tivoli [kurt94$_b$], the animation of a suggestion given by the system freezes if the user interrupts it by drawing a gesture. Although feedback design in Presto does not play long animation, long lengths of sounds are used to play back notes and rests, as described in Section 6.3.2 and Section 6.3.3. In these situations, it is better to give the user the flexibility to be able to interrupt the playback by drawing a gesture.

Presto can also let the user control fully the sounds played by giving them the option to switch the various sounds on or off. A dialog box can be used in Presto to enable users

to switch on or off the various sounds, so that the different feedback in the system can be evaluated.

6.4 Implementation

MusEd2 is the system developed to test the Presto2 gesture set that the author designed from Presto1. Most of the feedback issues that were discussed in Section 6.3 are added to MusEd2.

In MusEd2, a *ladder* with three ledger lines is added in the position of the cursor when the cursor is above or below the staff, as shown in Figure 6.2. Providing only three ledger lines is one of the limits in MusEd2 listed in Appendix B. The ladder moves with the cursor and when the cursor returns to the staff, the ladder disappears.

The option for switching the sound on and off is implemented in the menu in MusEd2, as shown in Figure 6.3. The user can also control various sounds by selecting the *sound* dialog box from the menu, as shown in Figure 6.4. Figure 6.5 shows the dialog box and the default sound options. This dialog box allows the user to choose certain sounds that would be played by the system. It can also be used to find out the impact of these sounds on music entry in the evaluation of Presto, which is discussed in Section 7.1. The volume of sound can be turned up or down by adjusting the speaker controls or the sound driver in the system.

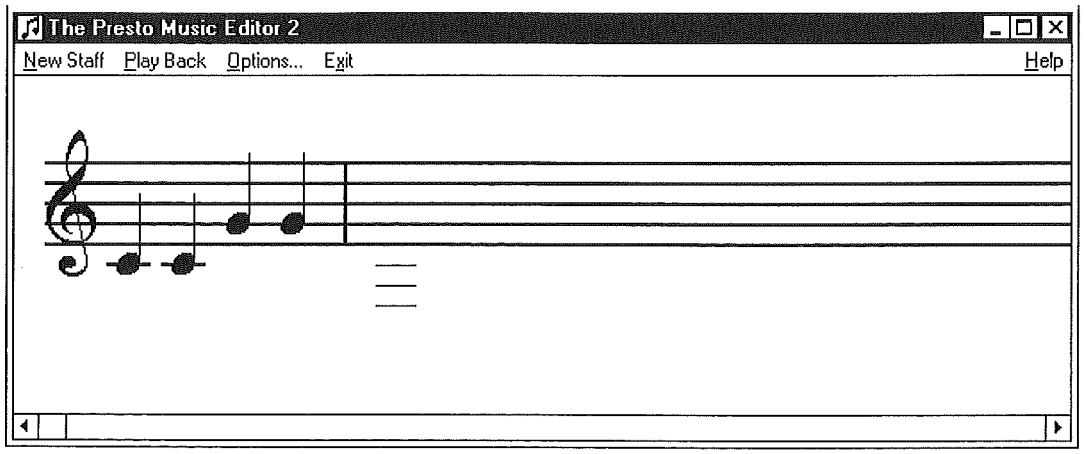


Figure 6.2: *Ladder* in MusEd2.

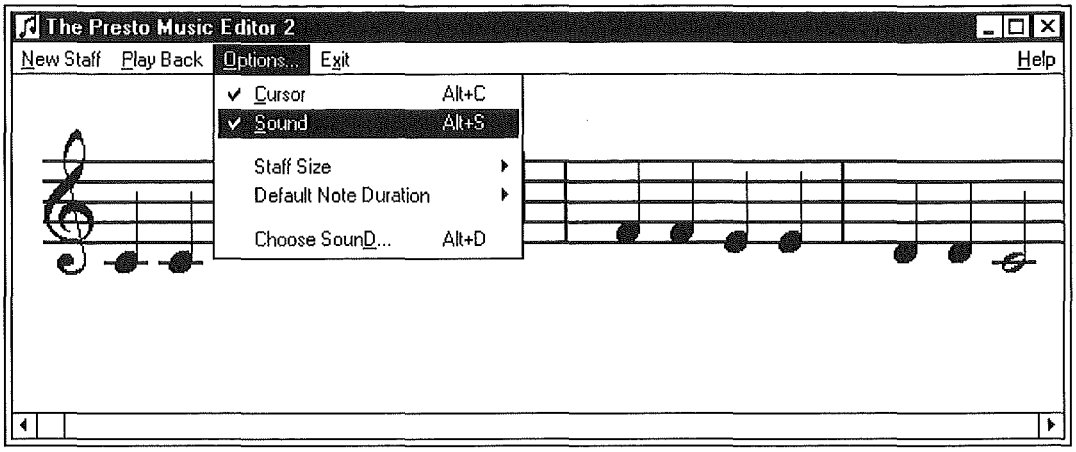


Figure 6.3: Sound option in MusEd2.

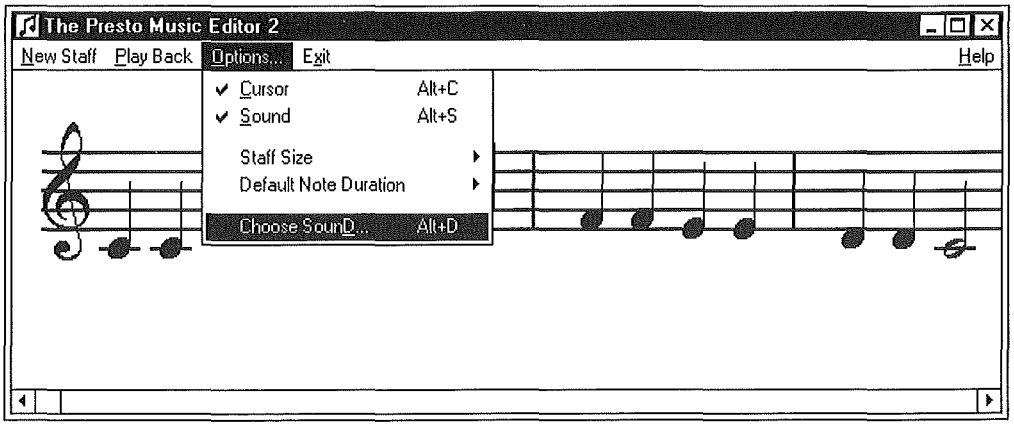


Figure 6.4: Get *sound* dialog box in MusEd2.

The simpler solution for visual feedback for entering and deleting symbols is implemented in MusEd2. Thus, when a symbol is inserted or deleted, a *star* appears temporarily around it, as illustrated in Figure 6.6. This image will stay for 0.096 second, which is equivalent to the duration of a semiquaver in MusEd2, then the star disappears for insertions or the symbol disappears for deletions.

In MusEd2, the user has the option of making notes play when a new note is inserted, and when the duration or the pitch of a note is changed. The note plays the *acoustic grand piano* from the *General MIDI instrument patch map* [mess98]. Similarly, the user can make rests play when a new rest is inserted and when the duration of a note is changed in MusEd2. The sound of the rest is the percussion instrument *high agogo* from

the *General MIDI percussion key map* [mess98]. The user can also choose to play the symbol when it is deleted. The sound of a deleted symbol is the *high wood block* from the percussion key map. These sounds usually have their own length.

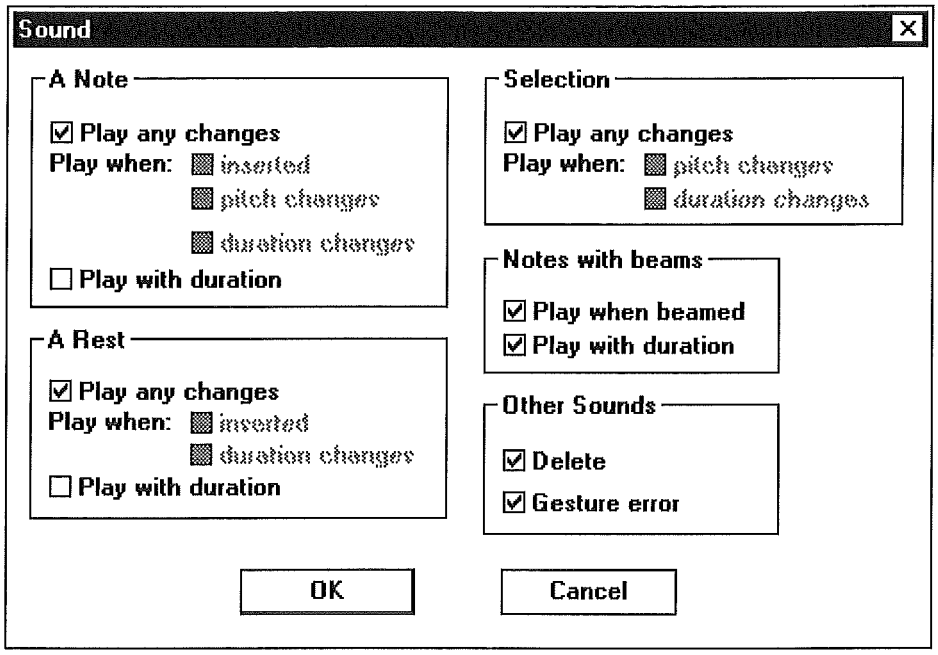


Figure 6.5: *Sound* dialog box in MusEd2.

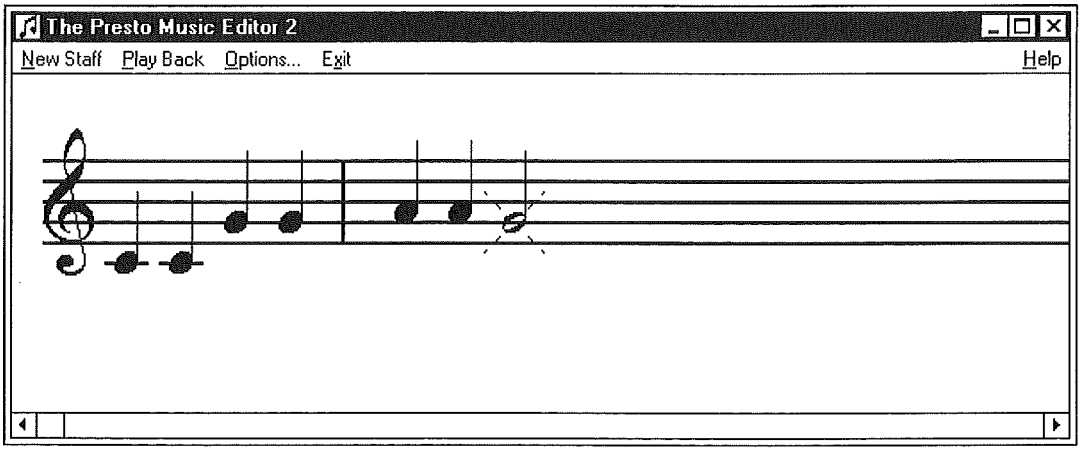


Figure 6.6: Inserted note with *star* in MusEd2.

A playback menu shown in Figure 6.7 is implemented in MusEd2. This command plays the notes lasting as long as their duration; rests are treated as silent notes. MusEd2 would scroll to the beginning of the score before it plays, and scroll along when the tracker gets near the edge of the window. MusEd2 can also play notes when the user selects and changes their pitch or duration, and when they are beamed. A *tracker*, which is the triangle below the staff in Figure 6.8, has been implemented to follow the playing symbol during playback. Playback can be interrupted by drawing another gesture, selecting a menu command, or pressing any key on the keyboard.

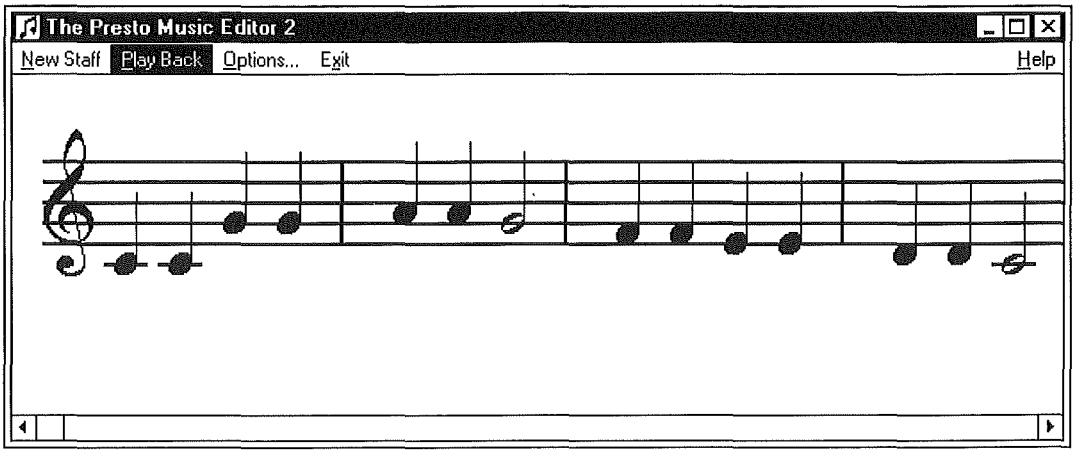


Figure 6.7: Playback option in MusEd2.

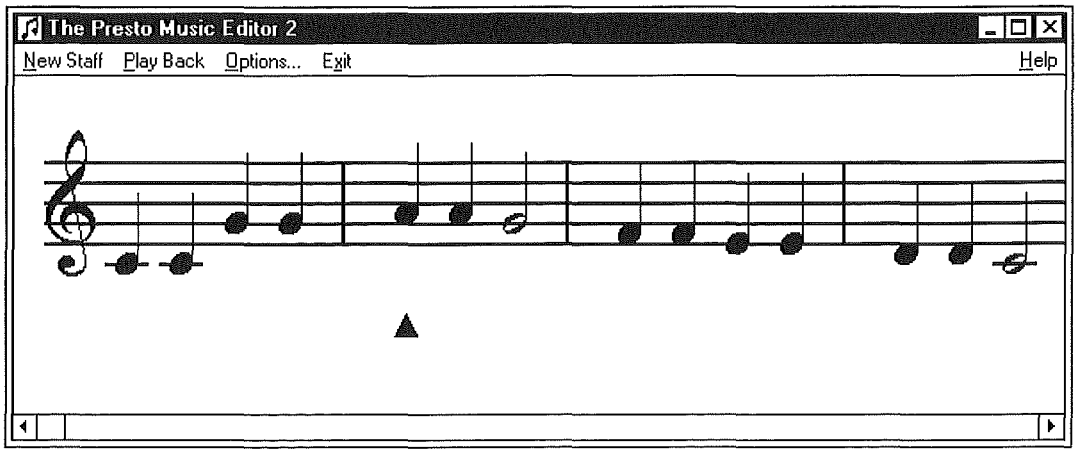


Figure 6.8: Playback *tracker* in MusEd2.

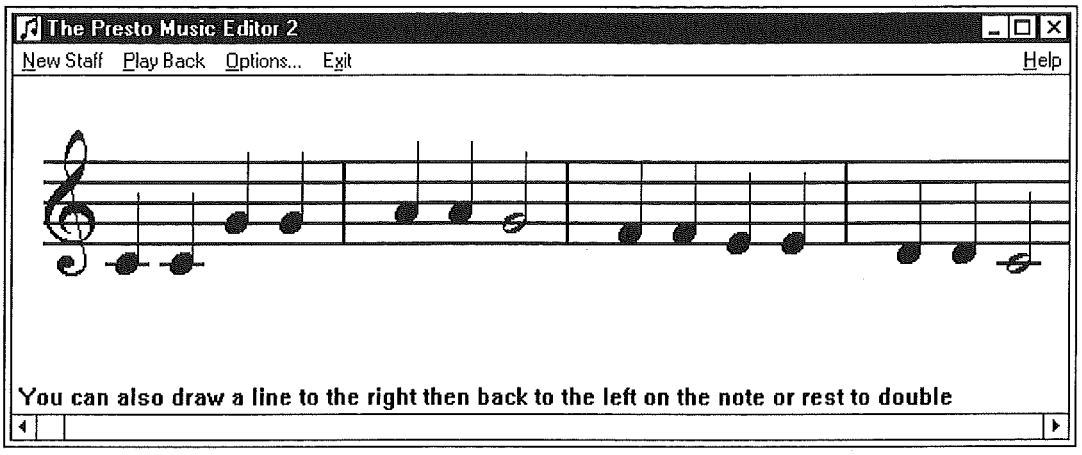


Figure 6.9: Help or error message at the bottom of the window in MusEd2.

The help message for suggesting an alternative gesture appears at the bottom of the window in MusEd2, as shown in Figure 6.9. There will be no sound to alert the user, because the changes in the note or the rest already produce a sound. Gesture error warnings also appear at the bottom of the window, but there is an option to play its sound, which is the *splash cymbal* from the percussion key map.

6.5 Summary

This chapter has briefly reviewed the importance of feedback, and looked specifically at visual and audio feedback. Visual and audio feedback is then designed and implemented into Presto to make the system easier to learn and use. This includes feedback when the user draws a note off the staff, inserts or deletes a musical symbol, and switches off various feedback; the system also needs to play back a selection, show help in alternative gestures, and send error messages.

Chapter 7

Evaluations

The main aims of Presto are to be fast and easy to use, thus it is important to test the speed and usability of Presto. The objectives of Presto are listed in Section 1.5. This chapter first describes these evaluations and next proposes further work on the Presto system. The usability evaluation was performed by a group of music students, and the speed evaluation was performed by three people involved in the research and trained to use Presto.

7.1 Usability

Eleven students from the School of Music at the University of Canterbury volunteered for the evaluation on the usability of Presto; nine of them were male and were studying in music composition, and two were female studying part-time in music. All the subjects were right-handed.

7.1.1 Method

The subjects were first introduced to the gesture set, Presto2, and the system, MusEd2 and were allowed to experiment with the system for 15 minutes. A list of the Presto symbols and their gestures, which is similar to Appendix A, was provided to help the subjects learn the gestures.

Next, they were given 25 minutes to complete a questionnaire on paper which appears in Appendix E. There are five sections in the questionnaire: gestures, editing features, beams, feedback and the complete editor. The full results along with subjects'

comments are in Appendix F. The sessions were not video-taped, so that the subjects would feel at ease when experimenting and answering questions about the system.

Subjects were also asked to briefly state their experiences in music and in using computers to help in the analysis of their answers. All eleven subjects have used computers before and eight of them use music editors. Only two subjects have used pen computers before: Subject 5 has used a pen tablet at work and Subject 8 owns a personal digital assistant.

7.1.2 Strengths of Presto

Overall, the subjects found Presto useful, but it requires a full set of features to be viable. Subject 6 found it "very ingenius", Subject 3 exclaimed "great", and Subject 9 said it is "interesting" and "definitely useful". Subject 1 even described Presto as a "fantastic tool for adding slurs, articulation marks, and dynamics to music before printing. It is so much easier and quicker and more flexible than a mouse." He thought it could be a "portable notepad" for musicians. Subject 2 described Presto as "what you are used to." Subject 5 has used pen computers before and thought Presto was "quite accurate compared to other pen systems." Subject 7 thought "it could be quite convenient for portable notation".

In addition to the positive response that the subjects gave, they were also enthusiastic about particular parts of the system, especially the playback function, the automatic scrolling function, and the option to choose a note to enter with the dot gesture.

Subject 1 found playing back the score in Presto "extremely useful especially if I [Subject 1] am writing or composing on the editor" and Subject 6 explained that "you can hear what you have written." In addition, the function was "absolutely compulsory" for Subject 3, "invaluable and really important" to Subject 7, and Subject 9 "definitely" needed it. Subject 1 also praised the automatic scrolling function as "good" and Subject 5 described it as a "useful feature." Subject 9 said he would need it and Subject 3 also said it is "very much needed and compulsory." Subject 6 found it "very helpful as you don't have to stop to do it." The option in the dot gesture was a "good idea" for Subject 2, "absolutely essential" for Subject 4, "very useful for initial input" for Subject 7, and "really good" for Subject 11.

All the subjects would use Presto to enter music into the computer. Subject 7 would use it "with a few alterations." Subject 6 would "most definitely" use it, Subject 11 "would like to have it at home," while Subject 10 hopes that it would be available commercially and commented that "I enjoyed using the system and finding out what it could do."

7.1.3 Weaknesses in Presto

Although Presto is generally successful, the subjects found several weaknesses that need addressing. These weaknesses are categorised into the following: inaccurate aiming, confusion with context-sensitive gestures, unintuitive gestures, inappropriate feedback, maintaining portability, and missing features. Possible solutions for these weaknesses are suggested in Section 7.3.

Parallex error

There were three subjects who had problems with aiming accurately on the staff of Presto. One subject kept entering notes at the wrong pitch, and the other two had difficulty deleting beams because they could not aim accurately on the beams.

Aiming on the pen computer could be affected by the angle that a user holds the pen, the angle that the user is viewing the tablet, and whether the user is aiming while looking at the pen tip or at the cursor. At the beginning of the evaluation, Subject 1 could not enter notes at the right pitch on Presto. This was because the angle that Subject 1 held the pen and the angle that he looked at the tablet might be different from those of the author, since the author had calibrated the pen on the computer before the evaluation according to her requirements. He was also aiming on Presto with the pen tip instead of the cursor. After Subject 1 calibrated the pen in the operating system, his aiming accuracy on Presto was nearly 100% and all the other subjects did not have this problem. Some of these subjects might also be aiming on Presto while looking at the cursor and not the pen tip. The cursor in the system is the picture of a feather and it may not be very useful for aiming on the staff, especially for the pitch of a note.

The second aiming problem happened when two subjects were deleting a beam. Subject 4 and Subject 8 found that they could not aim accurately on the beam. This meant that the tolerance for deleting a beam is too small. The tolerance cannot be any larger because the beam is a thin line and is often placed above or below another beam, thus there is a high risk of deleting a beam other than the intended one. However, the subjects still preferred a more sloppy area to delete beams.

Confusion with context-sensitive gestures

Five subjects either could not understand the concept of context-sensitive gestures or found it inconvenient to apply, especially with the gestures that draw a straight line to the left or to the right. These line gestures are used to add or remove durational dots on a note or a rest, halve a note or a rest, draw a beam over notes, and execute undo or redo depending on where the gestures are drawn.

Subject 1 and Subject 2 were confused with the gesture that adds a dot to a note or a rest and the gesture that halves a note or a rest; the note was often given an extra dot when they intended to halve it. After the author explained to them that they have to draw the halving gesture on the stem and not on the notehead, their input improved. However, they were still not satisfied with the gestures. Subject 4, Subject 9, and Subject 10 were unsure about the appropriate place to draw undo and redo gestures; it would be especially so in a multi-staves score. These subjects expressed that it was awkward for them to have the same gesture with different commands. They preferred to draw a gesture anywhere on the screen and the system would interprete it as the intended command.

Unintuitive gestures

There are some gestures that subjects found difficult to draw and were not intuitive. They are gestures that change the duration of a note or a rest, change the stem direction of a note, add or remove dots from a note or a rest, raise or lower a note, draw a broken beam, and select musical symbols.

Five subjects did not like to retrace the lines in gestures that consist of two lines. Subject 8 found the two-lined gestures for changing the duration of a note or a rest unintuitive. Subject 1 and Subject 11 suggested replacing them with one-lined left or right gestures; these gestures are already used to add or remove dots in Presto. Subject 1, Subject 4, and Subject 5 also found the gestures for changing the stem direction of a note unintuitive. Instead, they preferred to use the up and down gestures without retracing the line.

Subject 1 and Subject 2 did not find the add dot gesture intuitive. They suggested using the dot gesture to add a dot instead. These subjects and Subject 4 also found the remove dot gesture unnatural. They suggested using the delete gesture to remove the dot.

Subject 2 expected gestures that raise or lower a note would raise or lower it directly to the next line or space without accidentals instead of adding a sharp or a flat to the note, because he usually needs to raise or lower the note if he has entered it at the wrong pitch. Subject 2 would also use other gestures to enter an accidental and its note together, because musicians tend to draw on paper the accidental first before a note and not the other way round.

Subject 1 and Subject 3 found drawing broken beam gestures only "reasonably easy" due to the small tolerance provided for the gesture to be recognized. Subject 8 preferred to draw the broken beam gesture right through the stem of the note and not bother about placing the gesture on the left or right of the stem. To help the system differentiate between left and right broken beams, he would draw the gesture towards the direction that the broken beam points to; this is similar to an alternative solution for drawing broken beams described in Section 5.5.4.

Subject 7 found selection "quite difficult" and Subject 3 preferred to draw the selection gesture without pressing the button on the pen barrel and wanted notes that are touched by the gesture to be included too. Currently, the recognizer in Presto could only recognize straight lines in gestures. Thus, the gestures in the gesture set are restricted to straight lines, and the recognition of the enclosing gesture for selection needs the pen button to be pressed to be differentiated from line gestures.

Inappropriate feedback

Nine subjects found inappropriate features in Presto's feedback. These features are the *star* on an inserted or deleted symbol, the tracker during playback, the suggestions displayed by the help system, the error messages displayed at the bottom of the window, and reflecting the duration of a symbol in its sound.

Subject 1, Subject 4, and Subject 8 thought that the star that appeared temporarily on a inserted or deleted symbol was "not obvious enough to notice". Subject 2 suggested that gradually diminishing a deleted symbol would be "easier to follow;" this method is mentioned in Section 6.3.2.

Most subjects liked or needed the tracker during playback in Presto, but three subjects could follow the music along the score very well without it. Subject 4 preferred the tracker to move from bar to bar instead of symbol to symbol, or not to have it at all.

Although Subject 7 found the tracker useful, he thought it "might get annoying." Subject 10 found the tracker "not absolutely necessary."

The suggestions at the bottom of the window were not helpful for six subjects, but another subject wanted them to be more visible in the window. Subject 3, Subject 6, and Subject 9 ignored the suggestions displayed at the bottom of the window, and Subject 4 found the messages distracting. Subject 9 explained that if she has chosen a gesture to draw a symbol, she would continue to use that gesture, thus the information in the suggestions would be redundant. On the other hand, Subject 5 wanted the suggestions to "stand out more" and "be more obvious." Subject 10 said "it could have its uses but it is not absolutely necessary." Subject 7 also found the messages "useful especially at first, but it would get annoying," thus he suggested to have an option to turn it off.

In contrast, only Subject 6 did not need the error messages displayed at the bottom of the window; Subject 4 and Subject 5 found them "moderately useful," and all the other subjects found them more useful than the suggestions.

Playing the sound of a note or a rest that lasts as long as the duration in isolation was not meaningful to the subjects, thus only Subject 3 and Subject 5 selected this option.

Maintaining portability

Six subjects preferred Presto to operate without the mouse or the keyboard and found scrolling easiest using the pen on the scrollbar. Subject 1 and Subject 10 did not like to use the mouse. Subject 9 and Subject 10 preferred not to use the keyboard. Subject 1 explained that one hand would be holding the pen, and the other hand would be on the piano keyboard rather than on a mouse or the computer keyboard. In addition, Subject 10 would make full use of the pen, since it was already in their hands. He also thought that a pen computer should be portable and operate without a cumbersome keyboard.

Missing features

The subjects have provided a large list of features that are not in Presto, and only a selection are included in this section in these categories: missing functions in gestures, missing functions, missing feedback and missing symbols. The full list of features are listed in Appendix F.

- Missing functions in gestures

  Subjects wanted some gestures to perform certain functions that are missing in Presto. Six subjects wanted to change duration of beamed notes and four subjects wanted to change the stems of beamed notes which would flip beams to the other side of the notes. Subject 3 wanted to halve notes that are larger then crotchet by drawing a beam over them. Subject 3 and Subject 8 wanted to draw dotted notes with the dot gesture for compound time.

- Missing functions

  Some functions that the subjects desired are missing in Presto. For example, Subject 1 wanted to change a note into a rest directly instead of deleting the note and then inserting a rest; in this way the music would be kept in time. Subject 4 wanted to have more control over beams, in particular, to change the slope of the beams and the height of stems. Subject 4 also wanted to bookmark a place in the score and go to the next bar by pressing the *tab* key on the keyboard. Subject 2 wanted to change the speed of playback, Subject 4 and 11 wanted to play a selection of symbols, and Subject 4 wanted to choose the instrument for playback.

- Missing feedback

  Subjects required more feedback in Presto. After Subject 1 scrolled the score, he wanted to return to the position which he was working in. He suggested to flash the symbol that he was working on until he executes on another symbol. Subject 8 wanted the chosen note for dot gesture to be displayed on screen. Subject 2 wanted the played symbol to be highlighted with a different colour, and Subject 3 wanted a line moving continuously over the score in addition to the tracker during playback.

- Missing symbols

  Finally, there were many musical symbols that subjects found missing in Presto. Seven subjects wanted to draw slurs and ties in Presto, and six subjects wanted to add dynamic markings which include accents, crescendos, decrescendos, and those that indicate loudness and softness.

Symbol	Number
Filled note	216
Dot	4
Beam	119
Barline	40
Tail	1
Crotchet rest	5
Sharp	5
Total	390

Table 7.1: Number of symbols in the first forty bars of Haydn's Divertimento No. 15.

7.2 Input speed

This evaluation was conducted to find out the maximum performance in input speed using Presto, and to compare it with the speed of entering music onto paper and other methods for music input. Thus, the subjects in this evaluation have to be very familiar with the gestures in Presto. Three subjects tested the system; Subject 1 is the author of Presto and has tested Presto for one and a half years, Subject 2 is the previous author of Presto and had tested Presto two years ago for one year, and Subject 3 is the supervisor of these two authors. All three subjects have experience in music and are right-handed.

7.2.1 Method

The subjects copied an excerpt from Haydn's String Trio "Divertimento No. 15" (Robbins Landon edition) [hayd81] into Presto without using a list of the Presto2 symbols and their gestures, which is similar to Appendix A. Next, the subjects copied the same excerpt onto paper neatly such that they themselves, not others, could play from it. Timing for the music entry was done by a stop watch.

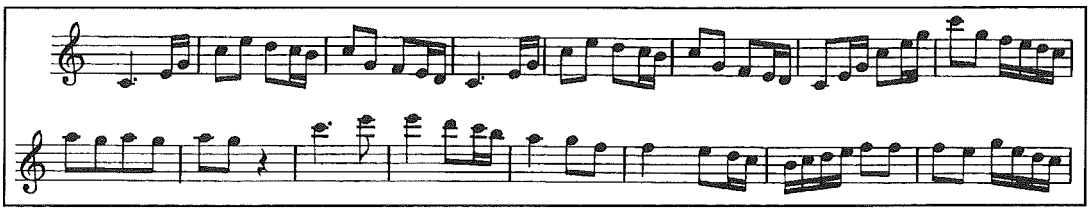


Figure 7.1: Bars 1 to 16 of Haydn's Divertimento No. 15 [hayd81].

	Presto			Handwriting		
	Total (min:sec)	**Ave / bar (sec)**	**Ave / symbol (sec)**	**Total (min:sec)**	**Ave / bar (sec)**	**Ave / symbol (sec)**
Subject 1	6:53	10.33	1.06	10:05	15.12	1.55
Subject 2	12:37	18.93	1.94	21:10	31.75	3.26
Subject 3	8:10	12.25	1.26	7:06	10.65	1.09
Average time	9:13	13.84	1.42	12:47	19.17	1.20

Table 7.2: Speed of Presto and handwriting by subjects.

The excerpt that was copied was the first forty bars of the first violin part of Haydn's "Divertimento No. 15". Results of speed performance were already available for copying these bars in Presto1, thus using the same piece of music would enable comparison between Presto1 and Presto2. Table 7.1 presents the distribution of symbols in these bars. Symbols that the system cannot process, such as the time signature, trills, ossias and grace notes, were removed. Figure 7.1 shows the first sixteen bars to be copied.

7.2.2 Results

The time taken for the subjects to copy the first forty bars is shown in Table 7.2; this table also presents the average time taken to draw one bar and one symbol. Subject 1, who is the author, was faster than the other two subjects and demonstrated the potential speed of Presto; she was the most experienced in Presto but has not copied music for about fifteen years. Subject 2 has not copied music for about seven years, while Subject

3 has been writing music in recent years. All three subjects could enter music with Presto faster than or nearly as fast as hand copying.

From Table 7.2, the average time for entering music in Presto2 is 72% of that by hand copying; Subject 3 was slower using Presto2 than hand copying by about one minute, but the quality of music writing by hand, shown in Figure 7.2, is not as legible as using Presto2, and the music itself is not in electronic form which can be easily manipulated in the computer. Subject 3 thought he could still improve his speed in Presto2.

Anstice [anst96$_a$] had estimated that the time for music entry by hand copying is 42% of the time using Optical Music Recognition (OMR) or 31% of the time using musical and computer keyboards. Using these figures, the average entry time of the three subjects on Presto2 is 30% of the time using OMR or 22% of that using keyboards. In other words, Presto2 can potentially be about three times as fast as OMR and more than four times as fast as keyboard methods, although the speed of Presto2 is based on only three subjects.

This estimation is faster than Anstice's [anst96$_a$], which states that Presto1 is at least three times as fast as other input methods. In addition, the best time achieved in Presto2, which is 6 minutes and 53 seconds, is about 27% faster than the best time in Presto1, which is 9 minutes and 24 seconds [anst96$_a$]. The reason that the speed in Presto has increased may be because its gesture design and recognition algorithm have improved significantly.

7.3 Further Work

This section discusses further work on Presto focusing on the weaknesses in Presto as discussed in Section 7.1.3, and on other ways to improve Presto.

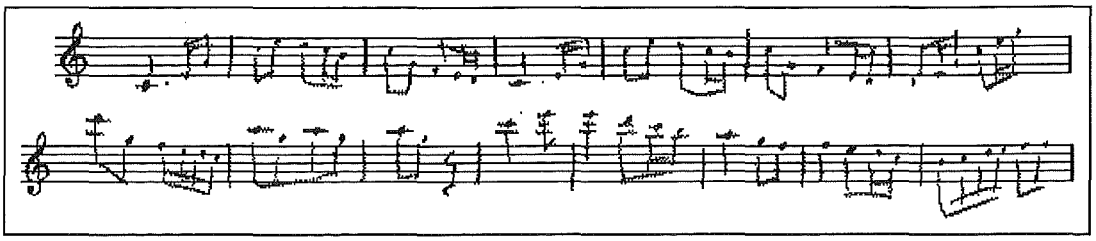


Figure 7.2: Music writing by hand.

7.3.1 Improving weaknesses in Presto

There are five categories of problems that users found in Presto—inaccurate aiming, confusion with context-sensitive gestures, unintuitive gestures, inappropriate feedback, maintaining portability, and missing features.

Parallex error

Further work in Presto can design features that help users aim more accurately in Presto. Users need to aim on the staff accurately to enter a note on certain pitch, change the properties such as the duration or pitch of a symbol, and delete a symbol. However, there are problems with entering notes on the wrong pitch and deleting beams, unless the user calibrate the pen. Thus, a special Presto calibration program can be designed to let users calibrate the pen on the staff. Another solution is to change the cursor to the picture of a notehead or a similar symbol that helps users to aim better on the staff.

The picture can also change to a cross-hair when users press the button on the pen to delete; this can help users to delete beams better. In addition, the tolerances for deleting the first and the last beams in a beamed group could be larger. The system can also adjust the tolerances for recognizing the beam gesture according to the speed that the gesture is drawn. In other words, a gesture that is drawn fast will be recognized in more sloppy tolerances than a slower gesture.

In addition to the problems stated in Section 7.1.3, further work on Presto can also highlight a symbol with a different colour when the cursor is on it before a gesture is drawn, so that users can tell that they are drawing on the intended symbol. This solution also helps to resolve the issues concerning context-sensitive gestures.

Confusion with context-sensitive gestures

In further work of Presto, users can be trained to draw gestures in terms of the respective context. Alternatively, these context-sensitive gestures can be changed to be less dependent in their context. The dot gesture rather than the line gesture can be used to add a durational dot to a symbol. However, this will conflict with the other functions of the dot gesture, especially the function that doubles a symbol. As a result, the doubling function must either be assigned to another gesture or be differentiated clearly from the other functions. A suggestion is to use the same dot gesture to execute both functions. For example, assuming there are only two durational dots on a note, the first gesture on

a crotchet without dots will add one dot, the second gesture will add the second dot, and the third will change the note into a minim without any dots. The durational dot can be removed by deleting it.

Highlighting the symbol with a different colour that the cursor touches on before they draw the gesture will be helpful to users. Undo and redo can use the gestures, as suggested in Section 4.2.2, which are similar to deletion, but they must not be misrecognized for each other. Assuming that the recognition algorithm will recognize more gestures, undo and redo can also be assigned to  and  respectively to resemble *u* for undo and *r* for redo. Alternatively, they can be available in the menu or as icons on the screen.

Unintuitive gestures

Further work in Presto can change the gestures that are not intuitive to ones that are more mnemonic to musical symbols. The gestures that change the duration of a note or a rest can be assigned to the left and right gestures; the left gesture halves and the right gesture doubles the symbol. The system can assign the dot gesture to add a durational dot to a note or a rest, and allow the delete gesture to remove durational dots. The up and down gestures can be assigned as gestures for changing the stem of a note up and down. Thus, the gestures that raise or lower a note have to be assigned other gestures; they could be  and  respectively. In addition, these gestures can raise or lower a note directly to the next line or space without adding accidentals. Since musicians tend to draw the accidental first before a note on the manuscript, new gestures can be designed to enter an accidental and its note together. A suggestion for drawing a crotchet with a sharp is  and for drawing a crotchet with a flat is  , assuming that Presto will recognize more gestures. Thus, users do not need to draw two or more gestures to obtain a note with an accidental.

Further work on broken beams is needed to find out if musicians prefer to draw a broken beam in the way that is designed in this research by drawing on the left or the right side of the stem or in the direction that the broken beam points to. The former allows the user to draw in either direction but restricts the area of the gesture, and the latter has a wider area to draw in but restricts the direction of the gesture.

If the recognizer in Presto uses a more efficient recognition algorithm and can recognize curved gestures, the gesture set can be expanded to include curved gestures and the

selection gesture can be recognized accurately. The selection gesture can also be implemented without pressing the button on the pen barrel and can include the notes that are touched by the gesture.

Inappropriate feedback

Further work in Presto can improve its feedback. Presto can gradually increase the darkness of an inserted symbol and keep it highlighted in a different colour until another symbols is executed, and gradually diminish a deleted symbol. During playback, users can choose to have the tracker move from symbol to symbol or bar to bar, or not have a tracker at all. The played symbol can also be highlighted with a different colour during playback. In addition, users can choose the instrument to play back the score on, and to change the speed of playback. Users can also play back a selection of the score or a few bars in the score.

Presto can inform users of the current position in the score by displaying a global time-line. Animation for the undo and redo commands will also be useful in Presto. Further work on the help system can expand it into an intelligent help system with icons and animation to better illustrate the suggestions. Tivoli [kurt94$_a$, pede95] demonstrated the potential of a help system using gestures. Figure 7.3 illustrates the help system expanding from the point of the gesture in Tivoli. The user can select one gesture, which is shown in Step 1 in Figure 7.4, to see its explanation and animation, as shown in Step 2, and follow the animation by drawing the gesture in Step 3, which will be executed in Step 4. The user can also interrupt the animation by drawing the gesture or other gestures. One drawback of this help system is the limited space on the screen to show large sets of possible gestures, thus a scrollbar may be needed. The user can also have the option to switch the help system and the error messages off.

Maintaining portability

Presto can continue to use mouse buttons and keys on the keyboard to execute commands, but the mouse and keyboard can be optional input devices; all the commands in Presto, such as the commands for pressing the *home* and *end* keys, can still be available using the pen either as icons or menus on screen or as gestures. However, assigning gestures to these commands that are rarely used will clutter the gesture set and make the gesture set harder for the user to remember.

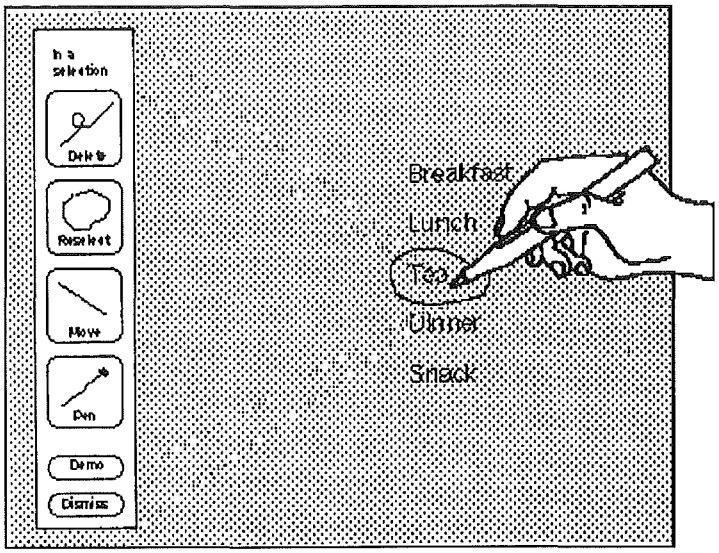


Figure 7.3: On-line help in Tivoli ([kurt94$_a$], fig. 6).

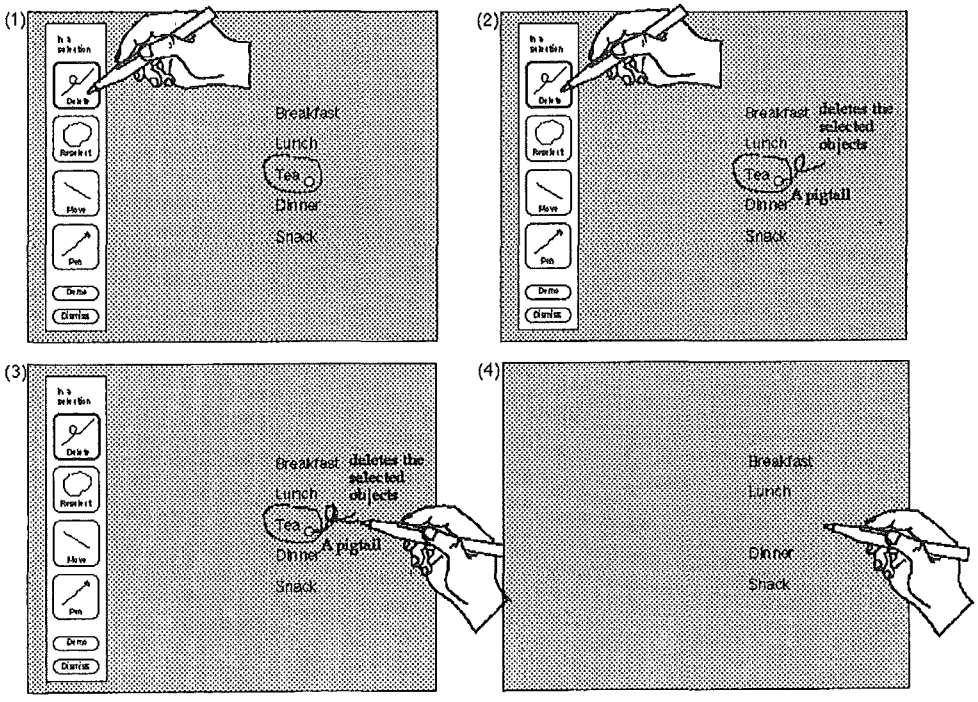


Figure 7.4: Demonstration of a gesture in Tivoli ([kurt94$_a$], fig. 7).

Missing features

A selection of missing features that are more popular and can be added into Presto are included in this section in these categories: missing functions in gestures, missing functions, missing feedback, and missing symbols.

- Missing functions in gestures

  Some gestures exist in Presto but do not do all the functions that users expect. Presto can allow users to use these gestures to change duration of beamed notes, change the stems of beamed notes, halve notes that are larger than crotchet by drawing a beam over them, add dotted notes into the option for the dot gesture, and provide multiple undo and redo commands.

- Missing functions

  A new gesture, such as  , can be designed to change a note into a rest. An alternative solution is to use the delete gesture; a single-dotted delete gesture on a note will turn the note into an equivalent rest, and a double-dotted delete gesture will delete the note completely. There can also be an option to automatically replace more than one rest with a equivalent rest. There can be control boxes on the two ends of a beam for changing the properties of the beam.

  Presto can allow users to scroll the score bar by bar by pressing on keys, such as *page up* and *page down*, or on icons on the screen. Presto can also have an option to let users mark places in the score with bookmarks and go to these places later in the session. In addition, Presto can let users change the speed of playback, play a selection of symbols, or choose the instrument for playback.

  Presto can implement more editing features like transpose, copy, cut, and paste a selection; print a selection or the whole score; and save a score to a standard file format, such as Musical Instrument Digital Interface [lang90, nola95], Notation Interchange File Format [niff95, gran96], or file formats of other music editors.

- Missing feedback

  Presto can provide a bookmark that automatically mark the last symbol that users are working on, so that users can always go back there by selecting that bookmark especially when they have scrolled the score to elsewhere. The chosen note for the

dot gesture can also be displayed on the screen. During playback, the played symbol can be highlighted with a different colour, and an optional line can move continously on the score. These can be provided in an option menu.

- Missing symbols

  Anstice [anst96$_a$] states that one of the most common symbols in music that is not included in Presto is the slur. A suggestion for the slur gesture could be ⌐, starting the gesture on the first note in the slur and ending on the last note. A tie can also be drawn in a similar way, since it is easily differentiated when the gesture starts on one note and ends on a consecutive note of the same pitch.

  Presto can allow the user to add lyrics in a window under the staff. In this window, a standard text recognizer provided with the pen computer can be used, and the lyrics can be automatically lined up with notes in the staff using standard algorithms. Currently, Presto accepts only monophonic music. Further work can allow chords to be added when the dot gesture is drawn above or below an existing note. Other additional musical symbols to include in Presto are dynamic markings, clef signs, time signature, and key signature.

7.3.2 Improving Presto

The system can eliminate redundant gestures for faster and more efficient entry. One example is to pre-draw bar lines in the system, so that the user can just fill in notes and rests in the bars. The user can still add and delete bar lines.

In the speed evaluation in this research, the time taken for music entry in Presto included the interval that subjects took to look at the excerpt to copy; this may be significant in the total input speed. Thus, a further evaluation on speed can require subjects to enter music that they have memorized to find the actual entry speed in Presto. In addition, extensive evaluation can also directly compare the speed of Presto to OMR and other methods [hewl$_2$94]. Testing can also be done on methods that combine input with OMR or musical keyboard, and editing in Presto.

The algorithm for calculating the slope of beams is simple and does not comply totally to the rules of Common Music Notation. Presto can use a more effective beam algorithm for drawing beams like the one proposed by Rader [rade96], or it can be

integrated into an existing music system such as Lime, which is described in Section 2.2.4, and use Lime to update the music representation on the screen.

Right and left-handed people may need different system interfaces [meye95, meye97]. Presently, the user is able to state which hand is holding the pen in the operating system, but there is no evaluation on right and left-handed input in this research. Moreover, all the subjects in both evaluations are right-handed. Therefore, further work on Presto can concentrate on the differences between using right and left hands to draw gestures. The issues on the ergonomics of pen computing [brow$_2$92, obor95], with regard to Occupational Overuse Syndrome in musicians, can be explored too.

A longer period will be needed for a better and more precise evaluation of Presto to find usability problems [niel92, niel93, wigg93]. The evaluation can be done using automated usability testing with systems such as Automated Usability Software (AUS) [chan$_1$97]. AUS takes several measures during evaluation, and can play back user's actions later without using a video camera.

7.4 Summary

This chapter has presented two sets of evaluations: usability and speed. Overall, Presto was well-accepted by the subjects in the usability evaluation, but it requires a full set of features to be viable. All of the subjects would use it to enter music into the computer with a few alterations. Another three subjects, including the author, tested Presto for speed, and showed that Presto can be four to five times faster than other methods to enter music. Further work on Presto was also discussed in detail.

Chapter 8

Conclusions

The Presto pen-based music input system was first designed and implemented by Anstice [anst96$_a$]. This thesis has further developed the system through another design iteration, looking at four parts—gestures, editing features, beams, and feedback. Each part was studied, designed, and implemented, forming a complete Presto system. Results of the evaluations of the system were also provided.

This research has made four contributions to the method of pen-based music input:

Accuracy. The research has improved the accuracy of recognizing gestures in Presto. The recognition algorithm in Presto was simplified to recognize gestures in four directions instead of eight directions. These gestures differ in terms of their shape, direction, and context. In this way, Presto can accept more sloppy gestures and recognize the gestures more accurately.

Gestures that consist of diagonal lines were redesigned, because the four-direction algorithm recognized only horizontal and vertical lines. These gestures include drawing a minim, drawing a crotchet with a fixed stem direction, changing the stem direction of a note, and changing the duration of a note or a rest.

Functionality. The research has enhanced the functionality of Presto by adding more editing and feedback functions into Presto. The thesis has investigated the issues of deletion, undo, selection, scrolling, and staff and cursor sizes in Presto. Deletion was redesigned to tap on the symbol while pressing the button on the pen barrel. Single undo and redo were also added into Presto. An encircle gesture was assigned for selection. Users could scroll the score in Presto by using the scrollbars, the arrow keys on the

keyboard, and the mouse. Presto2 could also automatically scroll the score if the user draws a gesture on the edge of the window. Three sizes of the staff were made available. Users could switch the pen cursor off.

In addition, this research has refined the beam gesture and allowed beams to be deleted in Presto. The visual rule follows the rules on beams in Common Music Notation and it is used as the paradigm for interpreting the beam gesture.

This thesis has shown that visual and audio feedback is useful in Presto. Feedback designed in Presto included feedback when the user drew a note that was not on the staff, and when the user inserted or deleted a musical symbol. The system played back the score, showed help in alternative gestures, and sent error messages too. The user could also switch the various types of feedback off.

Speed. The research has increased the speed of music entry in Presto. The recognition algorithm and gestures that were redesigned have not only improved accuracy, they have also resulted in faster entry speed. In addition, new gestures were designed for adding rests. The user could also choose different notes to draw with the dot gesture. The additional editing and feedback functions in Presto have increased the speed of music input in Presto by about 27%.

The speed evaluation on Presto indicated that the system has the potential for fast music input, especially when the users are experts in the system. In the evaluation, the subjects' input speeds were twice as fast as the speeds achieved by hand copying legibly onto paper.

The speed of music entry in Presto also compared favourably with the time taken to input music into the computer by other methods; music input in Presto could be three to more than four times faster than using other methods such as an Optical Music Recognition system or a keyboard-based system.

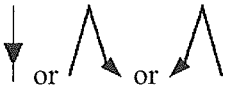
Usability. The research has extended the usability of Presto through improved accuracy, enhanced functionality, and increased speed; thus, making Presto easier to learn and use. From the usability evaluation on Presto, it seemed that pen-based interfaces are more natural and less intimidating than other graphical interfaces that use a menu or a mouse. Musicians may also be less resistant to these pen-based interfaces, and may find them easier to learn and use. In addition, the development of smaller and lighter low-cost pen computers will enable Presto to be portable for musicians.

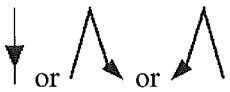
The contributions of Presto in this thesis confirm that pen-based music input is a useful technique for musicians. Presto can be an input method for any music notation system, or an editing interface for methods that are fast for input but slow for editing, such as entry by OMR or musical keyboard. Music transcription systems and computerised sight-singing tutors [mcna96$_a$, mcna96$_b$] can use Presto to enter new pieces of music into the computer. Presto can also be the natural writing interface for the *muse* which is a digital music stand for symphony musicians described in the beginning of this thesis [grae96$_a$, grae96$_b$].

Appendix A

Presto Gesture Set 2

This appendix presents a table of all the musical symbols and effects on them with their respective gestures available in Presto2, the gesture set that the author designed.

Symbol / effect	Gesture	Context
default note		Dot on pitch.
default note with upward stem		Start on pitch.
default note with downward stem		Start on pitch.
default note of double duration		Double dot on pitch.
quaver rest		Draw on staff.
crotchet rest		Draw on staff.
minim rest		Draw on staff.
semibreve rest		Draw on staff.
single bar-line		Draw on staff.

Symbol / effect	Gesture	Context
‖ double bar-line		Draw next to single bar-line.
Raise note (sharp & double sharp) *		Start on notehead.
Lower note (flat & double flat) *		Start on notehead.
Set stem upward *		Start on notehead.
Set stem downward *		Start on notehead.
Add durational dot *		Start on notehead or rest.
Remove durational dot *		Start on notehead or rest.
Double duration *		Start on notehead or rest.
		Dot on notehead or rest.
Halve duration *		Start on notehead or rest.
		Draw over the note or rest.
Beam (all notes or all rests)		Draw over notes or rests.
Delete symbol *		Dot with button on symbol.
Delete whole beam		Double dot with button on beam.
Delete double bar-line		Double dot with button on bar-line.
Undo		Draw out of staff.
Redo		Draw out of staff.
Selection		Draw out of staff.

* These can be executed in a selected group.

Appendix B

Presto Music Editor 2

This appendix briefly describes MusEd2, the pen-input music application that was prototyped to test the Presto2.

MusEd2 is implemented on an IBM-compatible Pentium PC running Microsoft Windows for Pen version 1.0, which is an extension to Microsoft Windows 3.1. The PC runs at 90 Mhz and has 32 Mb of RAM. The pen tablet is a Mutoh Video Tablet System 5 (VTS-5) with a 640-by-480 LCD screen on the high resolution digitizing tablet. MusEd2 is developed using Microsoft Visual C++ version 1.52, and is also ported to a Toshiba pen computer.

MusEd2 has these delimitations:

- it accepts and displays western common music notation;
- it uses melody scores, and the scores have treble clef and are single-voiced;
- it is not adaptable, that is, it does not learn gestures from the user;
- it produces readable and understandable music, and strict music engraving rules are not applied;
- it accepts and displays notes, rests, accidentals, durational dots, beams, bar-lines and double bar-lines, but it does not handle any other musical symbols;
- the highest note is between the third and the fourth ledger lines above the staff, and the lowest note is below the third and the fourth ledger lines below the staff;
- the maximum duration is the breve and its equivalent rest, and the minimum duration is the hemidemisemiquaver and its equivalent rest; and

- a note or a rest has a maximum of two durational dots.

MusEd2 is divided into two parts: a music gesture recognizer and a music input system. The recognizer is written in C, while the input system is written in C++.

B.1 Music gesture recognizer

This section describes the music gesture recognizer that is modified from the Windows handwriting recognizer provided with the compiler. The recognizer is compiled in the project *Musrec* using *musrec.mak*. It is also stored as a dynamic link library which is loaded by the system on demand.

B.1.1 *Musrec* files

These are the files included in the project *Musrec*:

File	Description
main.h	Header file for *musrec.c*.
musrec.c	Contains gesture recognizing functions.

B.1.2 *Musrec* functions

These are the functions in the *musrec.c* file:

Function	Description
dirline	Recognize direction of lines in four quadrants according to the directions of x and y dimensions, and allocate numbers to them. It has a total of four directions.
doRecogMus	Recognize gestures.

B.2 Music input system

This section describes the music input system. This system is compiled in the project *Music* using *music.mak*.

B.2.1 *Music* files

These are the files included in the project *Music*:

File	Description
barline.cpp	Contains member functions for class *Barline*.
drawmsc.cpp	Contains functions for drawing musical symbols.
drawmsc.hpp	Header file for *drawmsc.cpp*.
dummy.cpp	Contains member functions for class *Dummy*.
main.h	Header file for *musrec.c*.
mclass.hpp	Header file for *staff.cpp*, *barline.cpp*, *dummy.cpp*, *rest.cpp*, and *note.cpp*.
medit.cpp	Contains functions for Window applications.
medit.h	Header file for *medit.cpp*.
medit.rc	App Studio generated resource script for menus.
midisys.h	Header file for *midisys.dll*.
modify.cpp	Contains functions for executing gestures and commands.
modify.hpp	Header file for *modify.cpp*.
music.def	Contains explicitly exported functions.
note.cpp	Contains member functions for class *Note*.
pencur.cur	Cursor file.
presto.ico	Icon file.
resource.h	App Studio generated include file.
rest.cpp	Contains member functions for class *Rest*.
staff.cpp	Contains member functions for class *Staff*.
symbols.h	Contains musical symbols from Anastasia truetype font.

B.2.2 *Music* types

These are the classes in the *mclass.hpp* file:

Class	Description
Barline	A musical symbol in a *Staff* and a subclass of *Symbol*.
Dummy	A symbol in a *Staff*, a subclass of *Symbol*, and used at the beginning and end of each *Staff*.
Note	A musical symbol in a *Staff* and a subclass of *Symbol*.
Rest	A musical symbol in a *Staff* and a subclass of *Symbol*.
Staff	Contains a linked list of *Symbol*s.
Symbol	Superclass of *Barline*, *Dummy*, *Note* and *Rest*.

These are other types in the *mclass.hpp* and *drawmsc.hpp* file:

File	Name	Type	Description
drawmsc.hpp	*Size*	struct	Sizes of the staff and musical symbols.
mclass.hpp	*accidental*	enum	Accidental of a note.
mclass.hpp	*Beam*	struct	Beams of a note.
mclass.hpp	*cleftype*	int	Type of clef (bass or treble or none).
mclass.hpp	*Duration*	unsigned	Duration of a note or rest.
mclass.hpp	*keysigT*	int	Key signature of a note.
mclass.hpp	*midi_pitch*	unsigned	Midi version of the pitch.
mclass.hpp	*Pitch*	struct	Pitch of a note.
mclass.hpp	*symT*	char	Anastasia font of musical symbol.

B.2.3 *Music* representation structure

The data structure contains a linked list of staves and each staff contains a linked list of symbols, which can be barlines, notes, rests or dummies. When the system displays the structure, each staff draws itself and calls each symbol to draw itself. Figure B.1 illustrates this representation structure.

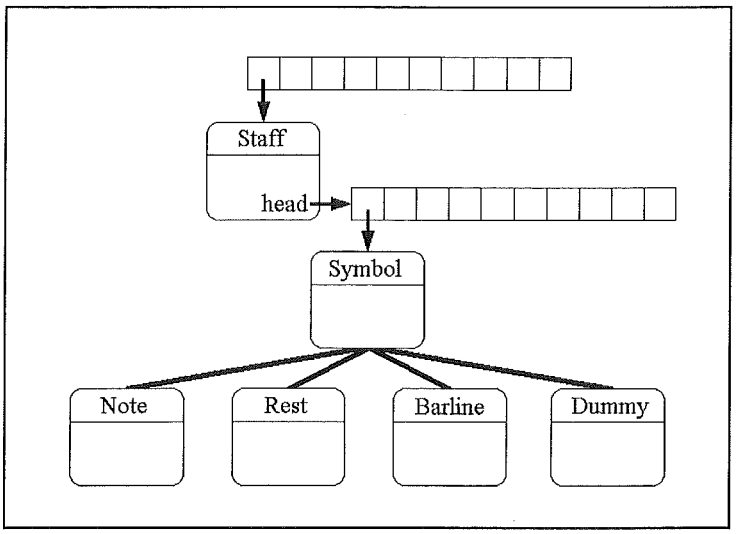


Figure B.1: Music representation structure in MusEd2.

B.2.4 *Music* functions

These are the functions in the *drawmsc.cpp* and *medit.cpp* files:

File	Function	Description
drawmsc.cpp	*drawThing*	Draws a musical symbol.
medit.cpp	*About*	Processes messages for "About" dialog box.
medit.cpp	*InitApplication*	Initialises window data and registers window class.
medit.cpp	*InitInstance*	Saves instance handle and creates main window.
medit.cpp	*MainWndProc*	Processes messages.
medit.cpp	*WinMain*	Initial entry point for the system. Calls initialisation functions and processes message loop.

These are the functions in the *modify.cpp* file:

Function	Description
doAddDot	Adds a dot to a note or a rest.
doAddNote	Adds a note.

Function	Description
doAddRest	Adds a rest.
doBeam	Adds a beam.
doChangeSize	Changes the sizes of the staff and the musical symbols.
doCleanStaff	Deallocates the memory of a staff.
doDAddNote	Adds a note that has double duration of a dot gesture's note.
doDDelete	Deletes a whole beam or a double bar-line.
doDelete	Deletes a musical symbol.
doDouble	Doubles the duration of a note or a rest.
doEndBar	Ends a bar.
doHalve	Halves the duration of a note or a rest.
doLower	Lowers the pitch of a note.
doRaise	Raises the pitch of a note.
doRedo	Redoes an undo.
doRemoveDot	Removes a dot from a note or a rest.
doStatus	Outputs an error message.
doSelAddDot	Adds dots to the selection.
doSelDelete	Deletes the selection.
doSelDouble	Doubles the selection.
doSelect	Highlights selected musical symbols.
doSelHalve	Halves the selection.
doSelLower	Lowers the selection.
doSelRaise	Raises the selection.
doSelRemoveDot	Removes the dots from the selection.
doSelTailUp	Draws the tails of the selection upwards.
doTailDown	Draws the tail of a note downwards.
doTailUp	Draws the tail of a note upwards.
doUndo	Undoes an action.
testDelete	Tests whether the musical symbol can be deleted.
testSelDelete	Tests whether the selection can be deleted.

B.3 System technical overview

WinMain is the initial entry point. First, it initializes the system by calling *InitApplication* and *InitInstance*, then it gets and processes messages from the user using *MainWndProc*. *MainWndProc* translates the messages and performs the appropriate actions. These actions include processing the commands from the main menu, recognizing gestures, manipulating the scrollbar, and performing the respective actions.

B.4 Gesture recognition rules

The gestures for different musical symbols are distinguished by the shape and size of the bounding box and the overall direction of the pen movement in both x and y directions. In addition, approximated sizes of x and y dimensions help to discriminate the gestures. Lastly, the pairs of vectors from the pen movement are quantised into four directions. Each direction is defined by a segment which subtends 90 degrees. These directions and segments are numbered in Figure B.2.

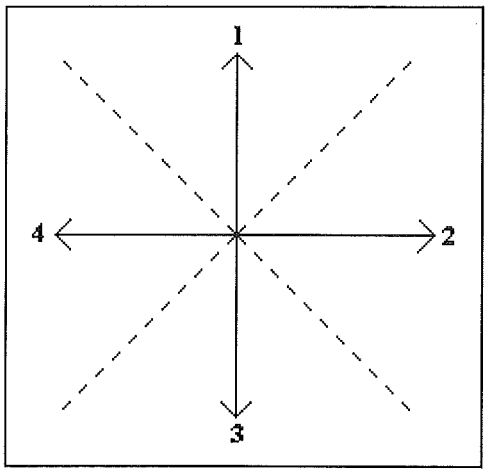


Figure B.2: Four directions of recognition.

B.4.1 Constants and definitions used

This is a table of the definitions and constants used to recognize the gestures:

Name	Definition	Value
small	Small size of *x* and *y* dimensions.	≤ 5 pixels.
left	Direction is 4.	If *initdir*, *middir*, & *enddir* = 4, then *true* ,else *false*.
right	Direction is 2.	If *initdir*, *middir*, & *enddir* = 2, then *true*, else *false*.
up	Direction is 1.	If *initdir*, *middir*, & *enddir* = 1, then *true*, else *false*.
down	Direction 3.	If *initdir*, *middir*, & *enddir* = 3, then *true*, else *false*.
numpts	Total number of pixels.	—
middle	Middle pixel.	*numpts* / 2.
initdir	If *small*, then direction of 0th to (*numpts*-1)th pixel, else direction of 5th to 8th pixel.	Return value of *dirline*.
enddir	If *small*, then direction of 0th to (*numpts*-1)th pixel, else direction of (*numpts*-8)th to (*numpts*-5)th pixel.	Return value of *dirline*.
meddir	If *small*, then direction of 0th to (*numpts*-1)th pixel, else direction of (*middle*-2)th to (*middle*+2)th pixel.	Return value of *dirline*.

B.4.2 Gestures

This is a table of gestures, their names, and their rules for recognition; they are checked against in the following order:

Gesture name	Rules	Gesture
syvDot	*small*	
syvUp	*up*	
syvDown	*down*	
syvRight	*right*	
syvLeft	*left*	
syvQuaverRest	*initdir* = 2 & *enddir* = 3	
syvCrotchetRest	*initdir* = 4 & *enddir* = 3	
syvTailUp	*initdir* = 1 & *enddir* = 3	
syvTailDown	*initdir* = 3 & *enddir* = 1	
syvDouble	*initdir* = 2 & *enddir* = 4	
syvHalve	*initdir* = 4 & *enddir* = 2	
syvSBreveRest	*initdir* = 3 & *enddir* = 2	
syvMinimRest	*initdir* = 1 & *enddir* = 2	
syvUnknown	Remaining unrecognized gestures	

B.5 Action rules

The gestures in Section B.4.2 are not translated directly into actions. In addition, not all actions are caused by gestures. This section lists the conditions for these actions and other factors that trigger the actions.

B.5.1 Gesture-triggered actions

This is a table of the gestures from Section B.4.3 and their appropriate actions from Section B.2.2 according to some conditions; if all the conditions are not satisfied, the system will treat the gesture as an error and display the error message:

Gesture name	Conditions and Actions
syvDot	*doDouble.* If *doDouble* failed, then *doAddNote*(default note).
syvUp	*doRaise.* If *doRaise* failed, then *doEndBar.*
syvDown	*doLower.* If *doLower* failed, then *doEndBar.*
syvRight	*doAddDot.* If gesture is outside staff, then *doRedo.* If *doAddDot* failed, then *doBeam.*
syvLeft	*doRemoveDot.* If gesture is outside staff, then *doUndo.* If *doRemoveDot* failed, then *doBeam.*
syvQuaverRest	*doAddRest*(quaver rest).
syvCrotchetRest	*doAddRest*(crotchet rest).
syvTailUp	*doTailUp.* If *doTailUp* failed, then *doAddNote*(default note with tail up).
syvTailDown	*doTailDown.* If *doTailDown* failed, then *doAddNote*(default note with tail down).
syvDouble	*doDouble.*
syvHalve	*doHalve.*
syvSBreveRest	*doAddRest*(semibreve rest).
syvMinimRest	*doAddRest*(minim rest).
syvUnknown	Return as unknown gesture.

A table of the musical symbols, effects on musical symbols, and their respective gestures in Presto2 is in Appendix A.

B.5.2 Other actions

This is a list of other actions that are triggered by non-gesture effects.

- If pen-down, then start recognition.
- If pen draw double dot, then *doDAddNote*.
- If the pen button is pressed
  ⇒ if *syvDot*, then *doDDelete*;
  ⇒ else *doSelect*.
- If left mouse button is clicked, then scroll window.
- If left mouse button is double-clicked:
  ⇒ if *shift* key is pressed, then *doDDelete*;
  ⇒ else if *shift* key is not pressed, then *doDAddNote*.
- If right mouse button is clicked, then *doDelete*.
- If right mouse button is double-clicked, then *doDDelete*.
- If mouse moves and left mouse button is pressed, then scroll window.
- If mouse moves and cursor is at area of ledger lines, then draw "ladder".
- If scrollbar is pressed, then scroll window.
- If *left* key is pressed, then scroll window to the left.
- If *right* key is pressed, then scroll window to the right.
- If *home* key is pressed, then scroll window to the beginning.
- If *end* key is pressed, then scroll window to the end.
- If *About Presto...* is selected from menu, then display *About* dialog box.
- If *Exit* is selected from menu, then close application.
- If *New Staff* is selected from menu, then *doCleanStaff*.
- If *Options...\Cursor* is selected from menu:

⇒ if cursor is on, then switch cursor off.

⇒ if cursor is off, then switch cursor on.

- If *Options...\Staff Size\small* is selected from menu, then *doChangeSize*(small).
- If *Options...\Staff Size\Medium* is selected from menu, then *doChangeSize*(medium).
- If *Options...\Staff Size\LARGE* is selected from menu, then *doChangeSize*(large).
- If *Options...\Default Note Duration\Quaver* is selected from menu, then *syvDot* draws quaver.
- If *Options...\Default Note Duration\Crotchet* is selected from menu, then *syvDot* draws crotchet.
- If *Options...\Default Note Duration\Minim* is selected from menu, then *syvDot* draws minim.
- If *Options...\Choose Sound...* is selected from menu, then display *Sound* dialog box.

Appendix C

Beam Survey Questionnaire

This appendix shows a sample of the beam survey questionnaire that was given to subjects in the beam survey discussed in Section 3.4.

QUESTIONNAIRE

Pen-based Music Editor: Beam survey

NOTE: You are invited to participate in the beam survey of the research project Pen-based Music Editor by completing the following questionnaire. The aim of this project is to explore the method of using pen-based computers to enter and edit music. The aim of this survey is to observe the way musicians draw beams on groups of notes on the music editor using a gesture similar to that on the manuscript. The questionnaire is anonymous, and you will not be identified as an informant without your consent. You may at any time withdraw your participation, including withdrawal of any information you have provided. By completing the questionnaire, however, it will be understood that you have consented to participate in the project, and that you consent to publication of the results of the project with the understanding that anonymity will be preserved.

The emphasis in this questionnaire is on drawing beams over groups of notes, and not on grouping notes rhythmically. Assume that you are in the midst of writing out a score such that illogical rhythmic grouping does not matter. Please note down any comments you would like to make about your answers.

Please briefly describe the amount of experience and training you have had in music (e.g. second year student in music).

.....

.....

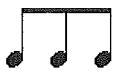
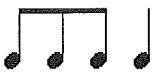
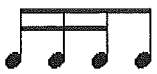
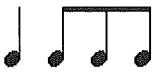
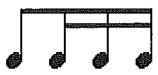
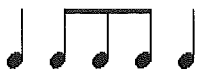
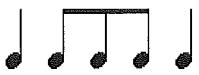
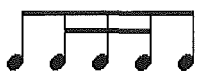
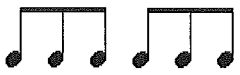
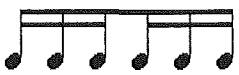
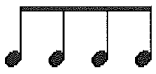
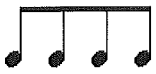
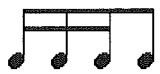
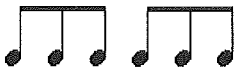
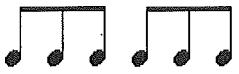
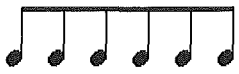
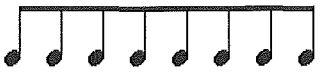
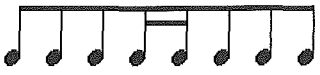
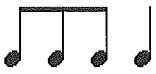
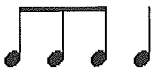
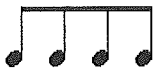
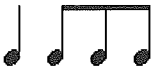
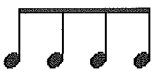
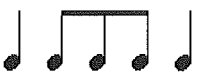
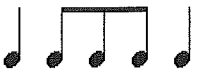
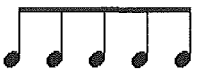
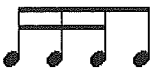
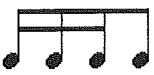
.....

.....

.....

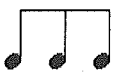
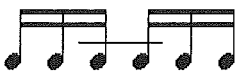
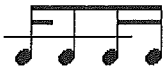
Exercise 1

Instructions: For each of the cases from (a) to (l), draw a line over the group(s) of notes in the "group of notes 1" column such that you will expect to get the result like that of the group of notes in "result" column. It may be necessary to draw more than one line in some of the cases. If you have another different action that you think would be equally suitable, draw it in the "group of notes 2" column. If you have more than two different actions, draw the others in different colours in the "group of notes 2" column. The first row gives an example where the horizontal line in column two represents a line that you have drawn.

Case	Group of notes 1	Group of notes 2	Result
E.g.			
a			
b			
c			
d			
e			
f			
g			
h			
i			
j			
k			
l			

Exercise 2

Instructions: For each of the cases from (a) to (h), draw beams over the group(s) of notes in the "result 1" column such that this is the result that you will expect to get from the group of notes and the line(s) in the "group of notes" column. It may be necessary to draw more than one beam in some of the cases. If you have another different result that you think would be equally suitable, draw it in the "result 2" column. If you have more than two different results, draw the others in different colours in the "result 2" column. The first row gives an example where the horizontal line in column three represents a beam that you have drawn.

Case	Group of Notes	Result 1	Result 2
E.g.			
a			
b			
c			
d			
e			
f			
g			
h			

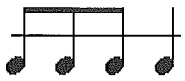
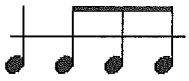
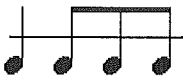
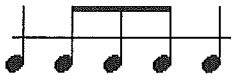
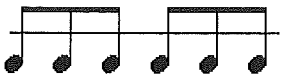
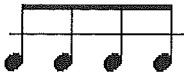
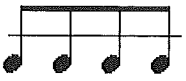
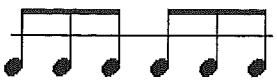
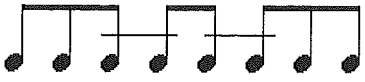
Thank you for your participation!

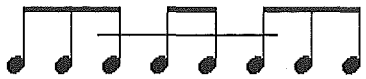
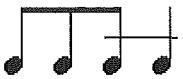
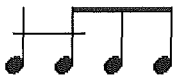
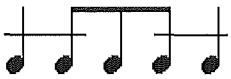
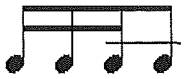
Appendix D

Beam Survey Model Answers

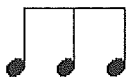
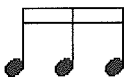
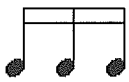
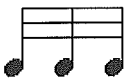
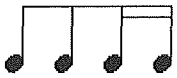
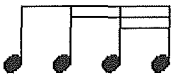
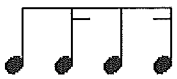
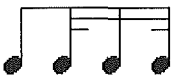
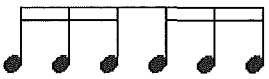
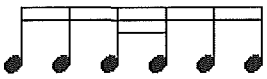
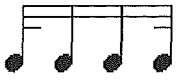
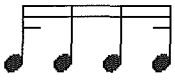
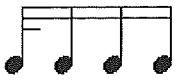
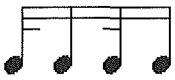
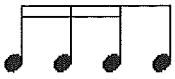
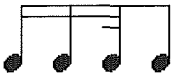
This appendix presents two sets of model answers for the beam survey questionnaire in Appendix C; one set uses the visual rule and the other set uses the logical rule. In the set of model answers that uses the logical rule, some answers are not possible to obtain by using only the beam gesture. The results of the beam survey is analysed in Section 5.4.

Exercise 1

Case	Visual rule	Logical rule
a		
b		
c		
d		not possible with beam gesture only
e		
f		not possible with beam gesture only
g		not possible with beam gesture only

Case	Visual rule	Logical rule
h		not possible with beam gesture only
i		not possible with beam gesture only
j		not possible with beam gesture only
k		not possible with beam gesture only
l		not possible with beam gesture only

Exercise 2

Case	Visual rule	Logical rule
a		
b		
c		
d		
e		
f		
g		
h		

Appendix E

Evaluation Questionnaire

This appendix shows a sample of the evaluation questionnaire that was given to subjects in the evaluation presented in Section 7.1.

QUESTIONNAIRE

Evaluation of the Pen-based Music Editor

NOTE: You are invited to participate in the evaluation of the research project Pen-based Music Editor by completing the following questionnaire. The aim of this project is to explore the method of using pen-based computers to enter and edit music. The aim of this evaluation is to find out the usefulness of the prototype system and its features to users. The questionnaire is anonymous, and you will not be identified as an informant without your consent. You may at any time withdraw your participation, including withdrawal of any information you have provided. By completing the questionnaire, however, it will be understood that you have consented to participate in the project, and that you consent to publication of the results of the project with the understanding that anonymity will be preserved.

Please briefly describe the amount of experience and training you have had in music (e.g. second year student in music).

.....

.....

.....

Have you used computers before? Yes / No

Have you used pen computers before? Yes / No

Have you used music editors before? Yes / No

Section 1. Gestures

Question 1. Please refer to Table 1 and tick the appropriate gesture(s). When you are drawing the gestures from the gesture set on the editor:

a. which effects/symbols do not do what they say in the gesture sheet?

b. which effects/symbols do you use most often?

c. which effects/symbols do you use the least?

d. which effects/symbols are easy to draw?

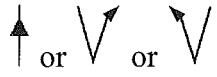
e. which effects/symbols are hard to draw?

f. what alternative gestures would you suggest for those that are hard to draw? (optional)

g. which gesture do you prefer to use to double or halve a note or rest?

h. which effects (that you have to draw with more than one gesture) would you like to be able to draw with only one gesture?

Table 1: Gesture table

Effect	Gesture	Question							
		a	**b**	**c**	**d**	**e**	**f**	**g**	**h**
Crotchet									
Crotchet (up stem)									
Crotchet (down stem)									
Minim									
Quaver rest									

Effect	Gesture	Question							
		a	**b**	**c**	**d**	**e**	**f**	**g**	**h**
Crotchet rest									
Minim rest									
Semibreve rest									
Single barline									
Double barline									
Raise note									
Lower note									
Stem up									
Stem down									
Add dot									
Remove dot									
Double (2 lines)									
Double (dot)									
Halve (2 lines)									
Halve (1 line)									
Beam									
Delete									
Delete beam									
Delete double barline									
Undo									
Redo									

Question 2. What other symbols would you most like to be able to draw in the editor? List three.

.....

Question 3. With regard to the option to choose what note to insert with the dot gesture:

a. how useful do you find the option?

.....

b. would you like to be able to insert rests with the dot gesture? Which one?

Yes No .....

Section 2. Editing Features

Question 4. When you delete a musical symbol:

a. how easy is it to press the button on the pen barrel?

.....

b. how easy do you find deletion?

.....

c. do you prefer to press the button on the pen barrel or press the *shift* key on the keyboard? pen button *shift* key

Question 5. How easy do you find undo?

.....

Question 6. How easy do you find redo?

.....

Question 7. How easy do you find selection?

.....

Question 8. How easy do you find changing the duration or the pitch of notes and rests with selection?

.....

Question 9. How useful do you find selection?

.....

Question 10. When you scroll the score:

a. how easy do you find scrolling with the scrollbar?

.....

b. how easy do you find scrolling with the mouse by pressing its left button?

.....

c. how easy do you find scrolling with the arrow keys on the keyboard?

.....

d. how useful do you find scrolling with the *home* and *end* keys on the keyboard?

.....

e. how useful do you find the automatic scrolling?

.....

f. do you want the place where the editor automatically scrolls to

stay the same, be nearer to the edge, or be further from the edge?

Question 11. Which size of the staff do you prefer?

small medium large

Question 12. Do you prefer the cursor to be

visible, or invisible?

Section 3. Beams

Question 13. Do you find it easy to understand how to draw beams on notes?

.....

Question 14. How easy do you find drawing beams on notes?

.....

Question 15. When you draw broken beams:

a. do you find it easy to understand how to draw broken beams?

.....

b. how easy do you find drawing broken beams?

.....

c. how easy do you find changing the direction of broken beams?

.....

Question 16. When you insert a note in a beamed group:

a. how easy do you find it?

.....

b. does the inserted note and its beams look like what you expect?

.....

Question 17. When you delete a note from a beamed group:

a. how easy do you find it?

.....

b. does the remaining beamed group looks like what you expect?

.....

Question 18. Would you like to be able to increase the duration of notes in the beamed groups? Yes No

Question 19. Would you like to be able to decrease the duration of notes in the beamed groups? Yes No

Question 20. How easy do you find deleting beams?

.....

Question 21. How easy do you find deleting whole beams?

.....

Section 4. Feedback

Question 22. Do you find the suggestions at the bottom of the screen useful?

.....

Question 23. Do you find the gesture error messages at the bottom of the screen useful?

.....

Question 24. Do you find it easy to insert notes that are not on the staff?

.....

Question 25. When you insert a note:

a. do you find the *star* around the note useful?

.....

b. do you find it easier with or without the sound?

.....

c. do you find it more useful with or without the duration?

.....

Question 26. When you insert a rest:

a. do you find the *star* around the rest useful?

.....

b. do you find it easier with or without the sound?

.....

c. do you find it more useful with or without the duration?

.....

d. do you like the sound that it plays?

.....

Question 27. When you delete a musical symbol:

a. do you find the *star* around the symbol useful?

.....

b. do you find it easier with or without the sound?

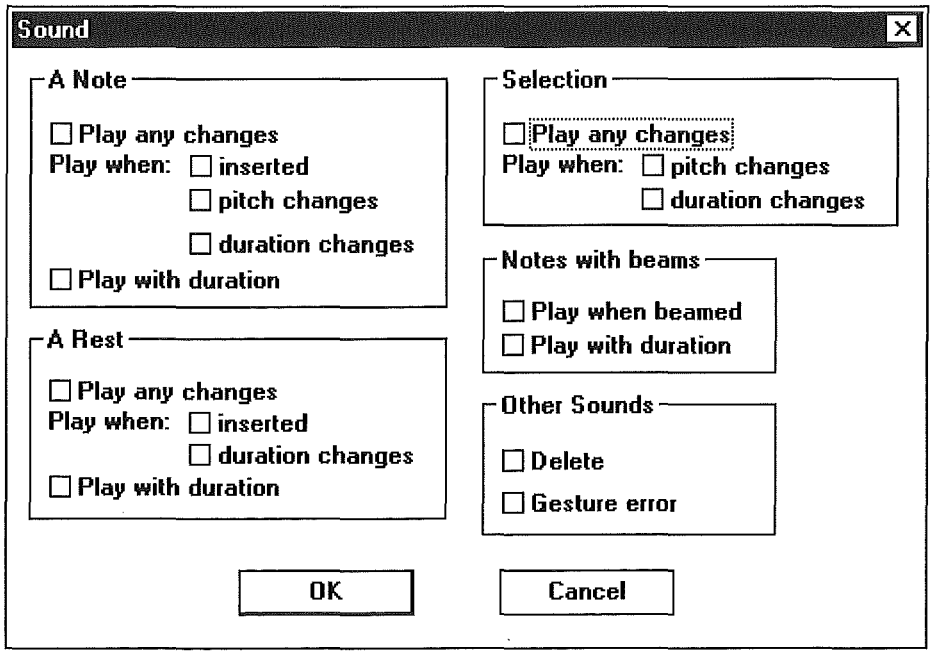
.....

c. do you like the sound that it plays?

.....

Question 28. For the following dialog box in the system, tick the options you would typically choose:

Figure 1: Sound dialog box.



Question 29. Do you find the playback of the whole score useful?

.....

Question 30. Do you find the tracker (the small triangle) useful for playback?

.....

Section 5. The complete editor

Question 31. Do you find the music editor useful?

.....

Question 32. Would you use the music editor for entering music into the computer?

.....

Question 33. What other comments do you have about the music editor?

.....

.....

.....

.....

Thank you for your participation!

Appendix F

Results of Evaluation Questionnaire

This appendix shows the results of the evaluation questionnaire that was given to 11 subjects in the evaluation presented in Chapter 7.

Number of subjects using:

	Yes	No
Computers	11 (100%)	0 (0%)
Pen computers	2 (18%)	9 (82%)
Music editors	8 (73%)	3 (27%)

Section 1. Gestures

Question 1. Please refer to Table 1 and tick the appropriate gesture(s).

a. which effects/symbols do not do what they say in the gesture sheet?

b. which effects/symbols do you use most often?

c. which effects/symbols do you use the least?

d. which effects/symbols are easy to draw?

e. which effects/symbols are hard to draw?

h. which effects (that you have to draw with more than one gesture) would you like to be able to draw with only one gesture?

Table 1: Gesture table

Effect/	Question					
Symbol	**a**	**b**	**c**	**d**	**e**	**h**
Crotchet (dot)	0 (0%)	8 (73%)	1 (9%)	7 (64%)	1 (9%)	0 (0%)
Crotchet (up stem)	0 (0%)	2 (18%)	3 (27%)	2 (18%)	1 (9%)	0 (0%)
Crotchet (down stem)	0 (0%)	2 (18%)	3 (27%)	2 (18%)	1 (9%)	0 (0%)
Minim	0 (0%)	5 (45%)	1 (9%)	6 (55%)	0 (0%)	0 (0%)
Quaver rest	0 (0%)	4 (36%)	0 (0%)	8 (73%)	0 (0%)	0 (0%)
Crotchet rest	0 (0%)	7 (64%)	0 (0%)	7 (64%)	1 (9%)	0 (0%)
Minim rest	0 (0%)	1 (9%)	2 (18%)	5 (45%)	0 (0%)	0 (0%)
Semibreve rest	0 (0%)	2 (18%)	2 (18%)	4 (36%)	0 (0%)	0 (0%)
Single barline	0 (0%)	9 (82%)	1 (9%)	8 (73%)	0 (0%)	0 (0%)
Double barline	0 (0%)	2 (18%)	4 (36%)	6 (55%)	0 (0%)	0 (0%)
Raise note	1 (9%)	6 (55%)	0 (0%)	5 (45%)	1 (9%)	4 (36%)
Lower note	0 (0%)	6 (55%)	0 (0%)	5 (45%)	1 (9%)	4 (36%)
Stem up	0 (0%)	3 (27%)	3 (27%)	4 (36%)	2 (18%)	1 (9%)
Stem down	0 (0%)	3 (27%)	3 (27%)	4 (36%)	2 (18%)	1 (9%)
Add dot	2 (18%)	7 (64%)	0 (0%)	5 (45%)	2 (18%)	0 (0%)
Remove dot	1 (9%)	2 (18%)	4 (36%)	3 (27%)	2 (18%)	0 (0%)
Double (2 lines)	0 (0%)	3 (27%)	2 (18%)	3 (27%)	1 (9%)	1 (9%)
Double (dot)	0 (0%)	6 (55%)	1 (9%)	3 (27%)	0 (0%)	0 (0%)
Halve (2 lines)	0 (0%)	3 (27%)	1 (9%)	2 (18%)	1 (9%)	1 (9%)
Halve (1 line)	2 (18%)	4 (36%)	1 (9%)	2 (18%)	2 (18%)	0 (0%)
Beam	0 (0%)	11 (100%)	0 (0%)	7 (64%)	0 (0%)	0 (0%)
Delete	0 (0%)	8 (73%)	0 (0%)	6 (55%)	0 (0%)	0 (0%)
Delete beam	1 (9%)	1 (9%)	0 (0%)	3 (27%)	1 (9%)	0 (0%)
Delete double barline	0 (0%)	1 (9%)	3 (27%)	4 (36%)	0 (0%)	0 (0%)
Undo	0 (0%)	5 (45%)	1 (9%)	3 (27%)	0 (0%)	0 (0%)
Redo	0 (0%)	3 (27%)	1 (9%)	3 (27%)	0 (0%)	0 (0%)

Question 1f. What alternative gestures would you suggest for the ones that are hard to draw?

- draw  or  to insert a crotchet, draw to insert a minim, draw  to insert a quaver, and draw  to insert a semibreve
- draw  and  to move a note to the next space or line without accidentals
- draw  to change stem down and draw  to change stem up
- draw  to insert a crotchet rest
- draw to insert a minim rest, and draw to insert a semibreve rest
- draw  to add a dot, and draw to delete a dot
- draw or hold pen down to change the duration of a note or a rest
- draw to add staccato

Question 1g. Which gesture do you prefer to use to double or halve a note or rest?

Double (2 lines) – 3 (27%)	Double (dot) – 8 (73%)
Halve (2 lines) – 5 (45%)	Halve (1 line) – 6 (55%)

Question 2. What other symbols would you most like to be able to draw in the editor?

Symbol	**Number**
slurs / ties	7 (64%)
dynamics (include loudness / softness, accents, crescendos, decrescendos)	6 (55%)
key and time signatures	3 (27%)
staccatos / tenutos	3 (27%)
triplets and the like	3 (27%)
chords	2 (18%)
other clefs	2 (18%)
repeat symbols	2 (18%)

Symbol	Number
text	2 (18%)
grace notes	1 (9%)
naturals	1 (9%)
ottavas	1 (9%)
pauses	1 (9%)

Question 3. With regard to the option to choose what note to insert with the dot gesture:

a. how useful do you find the option?

Useful – 10 (91%) Not useful – 1 (9%)

Comments:
- can also have dotted notes for compound time
- place options in toolbar
- show selected note or rest value on screen

b. would you like to be able to insert rests with the dot gesture? Which one?

Yes – 3 (27%) No – 8 (73%)

Comments:
- yes, crotchet and semibreve rests

Section 2. Editing features

Question 4. When you delete a musical symbol:

a. how easy is it to press the button on the pen barrel?

Easy – 11 (100%) Not easy – 0 (0%)

Comments:
- should have a light on the pen to indicate when button is pressed

b. how easy do you find deletion?

Easy – 10 (91%) Not easy – 1 (9%)

Comments:

- should also delete whole bar of symbols

c. do you prefer to press the button on the pen barrel or press the *shift* key on the keyboard?

Pen button – 10 (91%) *Shift* key – 1 (9%)

Comments:

- pen button especially when the other hand is on a piano keyboard
- pen is already in hand
- pen computer should be portable and operate without a keyboard
- *shift* key is more logical

Question 5. How easy do you find undo?

Easy – 9 (82%) Not easy – 2 (18%)

Comments:

- awkward to have the same gesture as other commands
- gesture has to be in the right place

Question 6. How easy do you find redo?

Easy – 9 (82%) Not easy – 2 (18%)

Question 7. How easy do you find selection?

Easy – 8 (73%) Not easy – 3 (27%)

Comments:

- intuitive and flexible
- prefers not to press the pen button
- should include notes that are touched by the gesture

Question 8. How easy do you find changing the duration or the pitch of notes and rests with selection?

Easy – 10 (91%) Not easy – 1 (9%)

Question 9. How useful do you find selection?

Useful – 11 (100%) Not useful – 0 (0%)

Comments:

- will use a lot especially for stems when there are interleaving notes

Question 10. When you scroll the score:

a. how easy do you find scrolling with the scrollbar?

Easy – 11 (100%) Not easy – 0 (0%)

Comments:

- very slow
- could use bookmarks to go to a certain place in the score

b. how easy do you find scrolling with the mouse by pressing its left button?

Easy – 6 (55%) Not easy – 5 (45%)

Comments:

- could move faster
- prefers not to use a mouse
- harder than the scrollbar

c. how easy do you find scrolling with the arrow keys on the keyboard?

Easy – 7 (64%) Not easy – 4 (36%)

Comments:

- could also scroll bar by bar
- better than the scrollbar
- slower than the scrollbar
- not preferable

d. how useful do you find scrolling with the *home* and *end* keys on the keyboard?

Useful – 10 (91%) Not useful – 1 (9%)

Comments:

- the best
- use a lot
- could use the *tab* key to go to the next bar
- should go to the end of the actual score rather than the end of the staff
- prefers it on screen

e. how useful do you find the automatic scrolling?

Useful – 11 (100%) Not useful – 0 (0%)

Comments:

- compulsory
- should keep current note flashing to help make sense of new surrounding
- should scroll sooner and smoother

f. do you want the place where the editor automatically scrolls to stay the same, be nearer to the edge, or be further from the edge?

Stay middle – 8 (73%) Less empty staff – 2 (18%) More empty staff – 1(9%)

Comments:

- empty staff should be ¼ of the window

Question 11. Which size of the staff do you prefer?

Small – 6 (55%) Medium – 3 (27%) Large – 2 (18%)

Comments:

- small, because can see more of staff
- multi-stave scores would need small

Question 12. Do you prefer the cursor to be

Visible – 8 (73%) Invisible – 3 (27%)

Comments:

- visible, because it is easier to place notes with it
- good to have the option
- invisible, because it is distracting

Section 3. Beams

Question 13. Do you find it easy to understand how to draw beams on notes?

Easy – 11 (100%) Not easy – 0 (0%)

Question 14. How easy do you find drawing beams on notes?

Easy – 11 (100%) Not easy – 0 (0%)

Comments:

- should also be able to control and change direction of beams
- should be able to draw a beam over many notes (larger than crotchets) to halve

Question 15. When you draw broken beams:

a. do you find it easy to understand how to draw broken beams?

Easy – 11 (100%) Not easy – 0 (0%)

b. how easy do you find drawing broken beams?

Easy – 9 (82%) Not easy – 2 (18%)

Comments:

- prefers to draw through stem
- should draw in the direction you want to go from stem

c. how easy do you find changing the direction of broken beams?

Easy – 9 (82%) Not easy – 2 (18%)

Question 16. When you insert a note in a beamed group:

a. how easy do you find it?

Easy – 11 (100%) Not easy – 0 (0%)

b. does the inserted note and its beams look like what you expect?

Yes – 11 (100%) No – 0 (0%)

Question 17. When you delete a note from a beamed group:

a. how easy do you find it?

Easy – 11 (100%) Not easy – 0 (0%)

b. does the remaining beamed group looks like what you expect?

Yes – 11 (100%) No – 0 (0%)

Question 18. Would you like to be able to increase the duration of notes in the beamed groups?

Yes – 6 (55%) No – 5 (45%)

Comments:
- save time

Question 19. Would you like to be able to decrease the duration of notes in the beamed groups?

Yes – 6 (55%) No – 5 (45%)

Question 20. How easy do you find deleting beams?

Easy – 10 (91%) Not easy – 1 (9%)

Comments:
- should be more sloppy in delete area
- cannot aim properly but can use undo

Question 21. How easy do you find deleting whole beams?

Easy – 10 (91%) Not easy – 1 (9%)

Comments:
- good gesture

Section 4. Feedback

Question 22. Do you find the suggestions at the bottom of the screen useful?

Useful – 4 (36%) Not useful – 7 (64%)

Comments:

- could be more obvious and have an option to turn it off
- would get annoying eventually
- ignored
- distracting
- would prefer to use one gesture only

Question 23. Do you find the gesture error messages at the bottom of the screen useful?

Useful – 10 (91%) Not useful – 1 (9%)

Comments:

- could be more obvious

Question 24. Do you find it easy to insert notes that are not on the staff?

Easy – 11 (100%) Not easy – 0 (0%)

Comments:

- could have more ledger lines

Question 25. When you insert a note:

a. do you find the *star* around the note useful?

Useful – 6 (55%) Not useful – 5 (45%)

Comments:

- could keep flashing until the next symbol is chosen
- could have option to turn it off
- did not notice

b. do you find it easier with or without the sound?

With – 9 (82%) Without – 2 (18%)

Comments:

- could have option to turn it off
- without, because it is distracting

c. do you find it more useful with or without the duration?

With – 3 (27%) Without – 8 (73%)

Comments:

- could not hear the difference

Question 26. When you insert a rest:

a. do you find the *star* around the rest useful?

Useful – 6 (55%) Not useful – 5 (45%)

b. do you find it easier with or without the sound?

With – 7 (64%) Without – 4 (36%)

c. do you find it more useful with or without the duration?

With – 2 (18%) Without – 9 (82%)

d. do you like the sound that it plays?

Yes – 9 (82%) No – 2 (18%)

Comments:

- should be the sound of a drum beam without tone (like deletion sound)

Question 27. When you delete a musical symbol:

a. do you find the *star* around the symbol useful?

Useful – 6 (55%) Not useful – 5 (45%)

Comments:

- should gradually diminish
- did not notice

b. do you find it easier with or without the sound?

With – 8 (73%) Without – 3 (27%)

c. do you like the sound that it plays?

Yes – 9 (82%) No – 2 (18%)

Question 28. For the following dialog box in the system, tick the options you would typically choose:

Play option	**Number**
play a note when it is inserted	10 (91%)
play a note when its pitch changes	10 (91%)
play a note when its duration changes	5 (45%)
play a note with its duration	2 (18%)
play a rest when it is inserted	8 (73%)
play a rest when its duration changes	6 (55%)
play a rest with its duration	2 (18%)
play selected symbols when their pitch changes	6 (55%)
play selected symbols when their duration changes	4 (36%)
play notes when they are beamed	4 (36%)
play beamed notes with their duration	3 (27%)
play deletion	8 (73%)
play gesture error	8 (73%)

Question 29. Do you find the playback of the whole score useful?

Useful – 11 (100%) Not useful – 0 (0%)

Comments:

- absolutely compulsory
- invaluable and really important
- useful especially if writing (composing) music
- should be able to choose instruments to play
- should be able to change the speed of playback
- should also play a selection

Question 30. Do you find the tracker (the small triangle) useful for playback?

Useful – 9 (82%) Not useful – 2 (18%)

Comments:

- note should also be highlighted
- should also have a constant meter (a line)
- may get annoying
- smoother motion and less flashing would be better
- prefers not to have, tracker on each bar is sufficient

Section 5. The complete editor

Question 31. Do you find the music editor useful?

Useful – 10 (91%) Not useful – 1 (9%)

Comments:

- require a full set of features
- very ingenius
- convenient for portable notation
- not useful at this stage of development

Question 32. Would you use the music editor for entering music into the computer?

Yes – 11 (100%) No – 0 (0%)

Comments:

- could be a portable notepad
- yes, with a few alterations

Question 33. What other comments do you have about the music editor?

- "this would be a fantastic tool for adding slurs, articulation marks, and dynamics to music before printing. It is so much easier and quicker and more flexible than a mouse."
- "I enjoyed using the system and finding out what it could do."
- "great"
- "interesting"
- got "hooked"
- "would like to have it at home"
- system is "cool" and hope that it will be available commercially
- viable system
- quite accurate compared to other pen systems
- flickering of the screen (when redrawing window) affects music entry
- need more staves in window
- be able to move symbols around the staff
- be able to beam rest
- be able to change a note into a rest
- halving and add dot gestures are confusing

Bibliography

[anst96$_a$] Anstice, Jamie. 1996. *A Pen-Input Computer Music Editing System.* MSc thesis. Christchurch, New Zealand: Department of Computer Science, University of Canterbury.

[anst96$_b$] Anstice, Jamie & Bell, Tim & Cockburn, Andy & Setchell, Martin. 1996. The design of a pen-based musical input system. *Proceedings of the Sixth Australian Conference on Computer-Human Interaction.* pp. 260–267. Los Alamitos, California: IEEE Computer Society.

[arch84] Archer, James E, Jr. & Conway, Richard & Schneider, Fred B. 1984. User recovery and reversal in interactive systems. *ACM Transactions on Programming Languages and Systems.* 6(1): 1–19. New York: ACM.

[assa86] Assayang, Gerard & Timis, Dan. 1986. A toolbox for music notation. *Proceedings of the 1986 International Computer Music Conference*, Den Hang. pp. 173–178.

[babe97] Baber, Christopher. 1997. *Beyond the Desktop: Designing and Using Interaction Devices.* Computers and People Series. San Diego: Academic Press.

[baec95$_a$] Baecker, Ronald & Grudin, Jonathan & Buxton, William A S & Greenberg, Saul. 1995. Touch, gesture, and marking. ch. 7. In *Readings in Human-Computer Interaction: Toward the Year 2000.* Baecker, Ronald & Grudin, Jonathan & Buxton, William A S & Greenberg, Saul (eds). pp. 469–482. San Francisco, California: Morgan Kaufmann.

[baec95$_b$] Baecker, Ronald & Grudin, Jonathan & Buxton, William A S & Greenberg, Saul. 1995. Vision, graphics design, and visual display. ch. 6. In *Readings in Human-Computer Interaction: Toward the Year 2000.* Baecker, Ronald & Grudin, Jonathan & Buxton, William A S & Greenberg, Saul (eds). pp. 411–423. San Francisco, California: Morgan Kaufmann.

[baec95$_c$] Baecker, Ronald & Small, Ian & Mander, Richard. 1995. Bringing icons to life. In *Readings in Human-Computer Interaction: Toward the Year 2000.* Baecker, Ronald & Grudin, Jonathan & Buxton, William A S & Greenberg, Saul (eds). pp. 444–449. San Francisco, California: Morgan Kaufmann.

[bain96] Bainbridge, David & Carter, Nicholas. 1996. Automatic reading of music notation. In *Handbook on Optical Character Recognition and Document Image Analysis.* Bunke, H & Wang, P S P (eds). Singapore: World Scientific.

[bain97] Bainbridge, David. 1997. *Extensible Optical Music Recognition.* PhD thesis. Christchurch, New Zealand: Department of Computer Science, University of Canterbury.

[barf93] Barfield, Lon. 1993. Feedback and errors. ch. 7. *The User Interface: Concepts & Design.* pp. 99–117. Wokingham, England: Addison-Wesley.

[belk94] Belkin, Alan. 1994. Macintosh notation software: Present and future. *Computer Music Journal.* 18(1): 53–69. Cambridge, Massachusetts: MIT.

[biel97] Bielawa, Herbert. 1997. Sibelius 7 music notation software. *Computer Music Journal.* 21(2): 99–105. Cambridge, Massachusetts: MIT.

[blos91] Blostein, Dorothea & Haken, Lippold. 1991. Justification of printed music. *Communications of the ACM.* 34(3): 88–99. New York: ACM.

[blos94] Blostein, D & Haken L. 1994. The Lime music editor: A diagram editor involving complex translations. *Software—Practice and Experience.* 24(3): 289–306. Chichester, England: Wiley Interscience.

[blys82] Bly, Sara. 1982. Presenting information in sound. *Proceedings of the Conference on Human Factors in Computer Systems.* pp. 371–375. Washington, District of Columbia: ACM.

[bobe97] Bob, Evil. 1997. *Mechanix of Music: Vocabulary Words* [online]. available URL: http://www.music-planet.com/theory/vocab.shtml

[brew93] Brewster, Stephen A & Wright, Peter C & Edwards, Alistair D N. 1993. An evaluation of earcons for use in auditory human-computer interfaces. *Human Factors in Computing Systems: "Bridges Between Worlds" INTERCHI '93 Conference Proceedings.* pp. 222–227. New York: ACM.

[brew94$_a$] Brewster, Stephen A & Wright, Peter C & Edwards, Alistair D N. 1994. A detailed investigation into the effectiveness of earcons. *Auditory Display: Sonification, Audification, and Auditory Interfaces.* Kramer, Gregory (ed.). pp. 471–498. Reading, Massachusetts: Addison-Wesley.

[brew94$_b$] Brewster, Stephen A & Wright, Peter C & Edwards, Alistair D N. 1994. The design and evaluation of an auditory-enhanced scrollbar. *Human Factors in Computing Systems: "Celebrating Interdependence" CHI '94 Conference Proceedings.* pp. 173–179. New York: ACM.

[brew95] Brewster, Stephen & Wright, Peter C & Edwards, Alistair D N. 1995. Parallel earcons: Reducing the length of audio messages. *International Journal of Human-Computer Studies.* 43(2): 153–175. London: Academic Press.

[brig$_1$87] Briggs, J S. 1987. Generating reversible programs. *Software—Practice and Experience.* 17(7): 439–453. Sussex, England: John Wiley & Sons.

[brig$_2$92] Briggs, Robert Owen & Beck, Brenda S & Dennis, Alan R & Carmel, Erran & Nunamaker, J F & Pfarrer, Richard. 1992. Is the pen mightier than the keyboard? *Proceedings of the IEEE.* 3: 201–210. New York: IEEE Computer Society.

[brod96$_a$] Brodhead, Thomas M. 1995–96. A beam study and a beam editor for computer-assisted music engraving. In *Computing in Musicology: An International Directory of Applications.* Hewlett, Walter B & Selfridge-Field, Eleanor (eds). vol. 10. pp. 203–206. Stanford, California: Center for Computer-Assisted Research in the Humanities.

[brod96$_b$] Brodhead, Thomas M. 1996. *Beams in Music Engraving.* Stanford, California: Center for Computer-Assisted Research in the Humanities.

[brow$_1$85] Brown, G. 1985. *Deluxe Music Construction Set.* San Mateo, California: Electronic Arts.

[brow$_2$92] Brown, Stephanie. 1992. *The Hand Book.* New York: Ergonome.

[buxt78] Buxton, William & Reeves, William & Baecker, Ronald & Mezei, Leslie. 1978. The use of hierarchy and instance in a data structure for computer music. *Computer Music Journal.* 2(4): 10–19. Menlo Park, California: People's Computer Company.

[buxt79] Buxton, William & Sniderman, Richard & Reeves, William & Patel, Sanand & Baecker, Ronald. 1979. The evolution of the SSSP score editing tools. *Computer Music Journal.* 3(4): 14–25. Menlo Park, California: People's Computer Company.

[buxt85] Buxton, William & Bly, Sara A & Frysinger, Steven P & Lunney, David & Mansur, Douglass L & Mezrich, Joseph J & Morrison, Robert C. 1985. Communicating with sound. *Human Factors in Computing Systems: "Mosaic of Creativity" CHI '95 Conference Proceedings.* pp. 115–119. New York: ACM.

[buxt86$_a$] Buxton, William. 1986. Chunking and phrasing and the design of human-computer dialogues. *Information Processing 86: Proceedings of the IFIP World Computer Congress.* pp. 495–499. Amsterdam: Elsevier Science.

[buxt86$_b$] Buxton, William. 1986. There's more to interaction than meets the eyes: Some issues in manual input. ch. 15. In *User Centered System Design: New Perspectives on Human-Computer Interaction.* Norman, Donald A & Draper, Stephen W (eds). pp. 319–337. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

[byrd86] Byrd, Donald. 1986. User interfaces in music-notation systems. *Proceedings of the 1986 International Computer Music Conference*, Den Haag. pp. 145–151.

[byrd94] Byrd, Donald. 1994. Music notation software and intelligence. *Computer Music Journal.* 18(1): 17–20. Cambridge, Massachusetts: MIT.

[cant71] Cantor, Don. 1971. A computer program that accepts common musical notation. *Computers and the Humanities.* 6(2): 103–109. Dordrecht: Kluwer Academic.

[cart88] Carter, N P & Bacon, R A & Messenger T. 1988. The acquisition, representation and reconstruction of printed music by computer: A review. *Computers and the Humanities.* 22(2): 117–135. New York: Pergamon Press.

[chan$_1$97] Chang, Elizabeth & Dillon, Tharam S. 1997. Automated usability testing. *Human-Computer Interaction: INTERACT '97.* pp. 77–84. London: Chapman & Hall.

[chan$_2$94] Chang, Larry & MacKenzie, I Scott. 1994. A comparison of two handwriting recognizers for pen-based computers. *Proceedings of CASCON '94.* pp. 364–371. Toronto: IBM Canada.

[chat96] Chatty, Stéphane & Lecoanet, Patrick. 1996. Pen computing for air traffic control. *Human Factors in Computing Systems: "Common Ground" CHI '96 Conference Proceedings.* pp. 87–94. New York: ACM.

[coda89] Coda's Finale for the Apple Macintosh and IBM PC. 1989. Products of interest. *Computer Music Journal.* 13(1): 83–84. Cambridge, Massachusetts: MIT.

[coda90$_a$] Coda Finale 2.0 music printing software for Apple Macintosh and IBM computers. 1990. Products of interest. *Computer Music Journal.* 14(1): 93–94. Cambridge, Massachusetts: MIT.

[coda90$_b$] Coda MusicProse and Finale 2.0 notation software for Apple Macintosh computers. 1990. Products of interest. *Computer Music Journal.* 14(2): 91–92. Cambridge, Massachusetts: MIT.

[coda93] Coda Finale 3.0. 1993. Products of interest. *Computer Music Journal.* 17(3): 92–93. Cambridge, Massachusetts: MIT.

[coop95] Cooper, Alan. 1995. Undo. ch. 30. *About Face: The Essentials of User Interface Design.* pp. 465–479. Danvers, Massachusetts: IDG Books Worldwide.

[craw83] Crawford, David & Zeeff, Jon. 1983. Gregory's Scribe: Inexpensive graphics for pre-1600 music notation. *Computer Music Journal.* 7(1): 21–24. Cambridge, Massachusetts: MIT.

[crot95] Crotty, Cameron. 1995. Graffiti 1.01. *Macworld: The Macintosh Magazine.* 12(5): 77. San Francisco, California: Macworld Communications.

[dalm78] dal Molin, Armando. 1978. A terminal for music manuscript input. *Computers and the Humanities.* 12(3): 287–289. New York: Pergamon Press.

[dann90] Dannenberg, Roger B. 1990. A structure for efficient update, incremental redisplay and undo in graphical editors. *Software—Practice and Experience.* 20(2): 109–132. Chichester: Wiley InterScience.

[dann93] Dannenberg, Roger B. 1993. Music representation issues, techniques, and systems. *Computer Music Journal.* 17(3): 20–30. Cambridge, Massachusetts: MIT.

[farr93] Farrand, A Brady & Rochkind, Mare & Chauvet, Jean-Marie & Tognazzini, Bruce & Smith, David C. 1993. Common elements in today's graphical user interfaces: The good, the bad, and the ugly. *Human Factors in Computing Systems: "Bridges Between Worlds" INTERCHI '93 Conference Proceedings.* pp. 470–473. New York: ACM.

[fins82] Finseth, Craig A. 1982. Managing words: What capabilities should you have with a text editor? *Byte.* 7(4): 302–310. Peterborough, New Hampshire: McGraw-Hill.

[fran92] Frankish, Clive & Morgan, Pam & Noyes, Jan. 1994. Pen computing: Some human factors issues. *IEE Colloquium on 'Handwriting and Pen-Based Input'*. pp. 5/1–3. London: Institute of Electrical Engineers.

[fran95] Frankish, Clive & Hull, Richard & Morgan, Pam. 1995. Recognition accuracy and user acceptance of pen interfaces. *Human Factors in Computing Systems: "Mosaic of Creativity" CHI '95 Conference Proceedings*. pp. 503–510. New York: ACM.

[free95] Freed, Adrian. 1995. Improving graphical user interfaces for computer music applications. *Computer Music Journal.* 19(4): 4. Cambridge, Massachusetts: MIT.

[gali96] Galitz, Wilbert O. 1997. Provide effective messages, feedback, guidance, and language translation. Step 11. *The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques.* pp. 540–590. New York: John Wiley & Sons.

[gave91] Gaver, William W & Smith, Randall B & O'Shea, Tim. 1991. Effective sounds in complex systems: The ARKola simulation. *Human Factors in Computing Systems: "Reaching Through Technology" CHI '91 Conference Proceedings*. pp. 84–90. New York: ACM.

[gave93] Gaver, William W. 1993. Synthesizing auditory icons. *Human Factors in Computing Systems: "Bridges Between Worlds" INTERCHI '93 Conference Proceedings*. pp. 228–235. New York: ACM.

[gave94] Gaver, William W. 1994. Using and creating auditory icons. *Auditory Display: Sonification, Audification, and Auditory Interfaces*. Kramer, Gregory (ed.). pp. 417–446. Reading, Massachusetts: Addison-Wesley.

[gave95] Gaver, William W & Smith, Randall B. 1995. Auditory icons in large-scale collaborative environments. In *Readings in Human-Computer Interaction: Toward the Year 2000.* Baecker, Ronald & Grudin, Jonathan & Buxton, William A S & Greenberg, Saul (eds). pp. 564–569. San Francisco, California: Morgan Kaufmann.

[gidd72] Giddings, B J. 1972. Alpha-numerics for raster displays. *Ergonomics*. 15(1): 65–72. London: Taylor & Francis.

[gold91] Goldberg, David & Goodisman, Aaron. 1991. Stylus user interfaces for manipulating text. *Proceedings of the Fourth Annual ACM Symposium on User Interface Software and Technology*. pp. 127–135. New York: ACM.

[gold93] Goldberg, David & Richardson, Cate. 1993. Touch-typing with a stylus. *Human Factors in Computing Systems: "Bridges Between Worlds" INTERCHI '93 Conference Proceedings*. pp. 80–87,520. New York: ACM.

[gomb77] Gomberg, David A. 1977. A computer-oriented system for music printing. *Computers and the Humanities*. 11(2): 63–80. New York: Pergamon Press.

[gonz96] Gonzalez, Cleotilde. 1996. Does animation in user interfaces improve decision making? *Human Factors in Computing Systems: "Common Ground" CHI '96 Conference Proceedings*. pp. 27–34. New York: ACM.

[gour86] Gourlay, John S. 1986. A language for music printing. *Communications of the ACM*. 29(5): 388–401. New York: ACM.

[grae96$_a$] Graefe, Christopher & Wahila, Derek & Maguire, Justin & Dasna, Orya. 1996. Designing the muse: Digital music stand for the symphony musician. *Human Factors in Computing Systems: "Common Ground" CHI '96 Conference Proceedings*. pp. 27–34. New York: ACM.

[grae96$_b$] Graefe, Christopher & Wahila, Derek & Maguire, Justin & Dasna, Orya. 1996. muse: Digital music stand for symphony musicians. *Interactions*. 3(3): 27–35. New York: ACM.

[graf96] Newton software review: Graffiti 2.0 [online]. 1996. *Pen Computing Magazine Issue 8*. Pen Computing Magazine. available URL: http:// pencomputing.com/PCM_8/P8_graffiti_ 2.0_review.html

[gran96] Grande, Cindy & Belkin, Alan. 1996. The development of the notation interchange file format. *Computer Music Journal*. 29(4): 33–43. Cambridge, Massachusetts: MIT.

[hake93] Haken, Lippold & Blostein, Dorothea. 1993. The Tilia music representation: Extensibility, abstraction, and notation contexts for the Lime music editor. *Computer Music Journal.* 17(3): 43–58. Cambridge, Massachusetts: MIT.

[hake97] Haken, Lippold & Blostein, Dorothea. *Music notation: Lime* [online]. available URL: http://www.ersmedia.com/liblime2.htm

[harr95] Harrison, Susan M. 1995. A comparison of still, animated or nonillustrated on-line help with written or spoken instruments in a graphical user interface. *Human Factors in Computing Systems: "Mosaic of Creativity" CHI '95 Conference Proceedings.* pp. 82–89. New York: ACM.

[hayd81] Haydn, Joseph. 1981. Divertimento No. 15 C-Dur für 2 Violinen und Violoncello Hob. V:16. *Diletto Musicale DM915: Joseph Haydn Streichtrios.* Landon, H C Robbins (ed.). Vienna: Doblinger.

[hela88] Helander, Martin. 1988. Input devices. ch. 22. In *Handbook of Human-Computer Interaction.* Helander, Martin (ed.). pp. 501–504. Amsterdam: Elsevier Science.

[henr90] Henry, Tyson R & Hudson, Scott E & Newell, Gary L. 1990. Integrating gesture and snapping into a user interface toolkit. *Proceedings of the Third Annual ACM SIGGRAPH Symposium on User Interface Software and Technology.* pp. 112–122. New York: ACM.

[heus87] Heussenstamm, George. 1987. *The Norton Manual of Music Notation.* New York: W W Norton.

[hewl$_1$95] Hewlett Packard. 1995. *HP OmniGo 100 Quick Start and User's Guide.* California: Hewlett Packard.

[hewl$_2$94] Hewlett, Walter B & Selfridge-Field, Eleanor. 1993–94. How practical is optical music recognition as an input method. In *Computing in Musicology: An International Directory of Applications.* Hewlett, Walter B & Selfridge-Field, Eleanor (eds). vol. 9. pp. 159–166. Stanford, California: Center for Computer-Assisted Research in the Humanities.

[ibmm94] IBM. 1994. *IBM Dictionary of Computing*. McDaniel, George (ed.). New York: IBM & McGraw-Hill.

[itoh90] Itoh, Kazunori & Yonezawa, Yoshimichi. 1990. Support system for handwriting characters and drawing figures for the blind using feedback of sound imaging signals. *Journal of Microcomputer Applications*. 13(2): 177–183. London: Academic Press.

[jack97] Jackson, David & Fovargue, Andrew. 1997. The use of animation to explain genetic algorithms. *SIGCSE Bulletin*. 29(1): 243–247. New York: ACM.

[kimu96] Kimura, Takayuki Dan. 1996. A pen-based prosodic user interface for schoolchildren. *IEEE Multimedia*. 3(4): 48–55. Los Alamitos, California: IEEE Computer Society.

[klau93] Klauer, B & Waldschmidt, K. 1993. An object-oriented pen-based recognizer for handprinted characters. *Computer Analysis of Images and Patterns: 5th International Conference, CAIP '93*. pp. 586–593. Berlin: Springer-Verlag.

[kram94] Kramer, Gregory. 1994. An introduction to auditory display. *Auditory Display: Sonification, Audification, and Auditory Interfaces*. Kramer, Gregory (ed.). pp. 1–77. Reading, Massachusetts: Addison-Wesley.

[krum90] Krummel, D W & Sadie, Stanley (eds). 1990. *Music Printing and Publishing*. The New Grove Handbooks in Music. New York: W W Norton.

[kurt91$_a$] Kurtenbach, Gordon & Buxton, Bill. 1991. GEdit: A test bed for editing by contiguous gestures. *SIGCHI Bulletin*. 23(2): 22–26. New York: ACM.

[kurt91$_b$] Kurtenbach, Gordon & Buxton, William. 1991. Issues in combining marking and direct manipulation techniques. *Proceedings of the Fourth Annual ACM Symposium on User Interface Software and Technology*. pp. 137–144. New York: ACM.

[kurt93] Kurtenbach, Gordon & Buxton, William. 1993. The limits of expert performance using hierarchic marking menus. *Human Factors in Computing Systems: "Bridges Between Worlds" INTERCHI '93 Conference Proceedings*. pp. 482–487. New York: ACM.

[kurt94$_a$] Kurtenbach, Gordon & Buxton, William. 1994. User learning and performance with marking menus. *Human Factors in Computing Systems: "Celebrating Interdependence" CHI '94 Conference Proceedings*. pp. 258–264. New York: ACM.

[kurt94$_b$] Kurtenbach, Gordon & Moran, Thomas P & Buxton, William. 1994. Contextual animation of gestural commands. *Proceedings of Graphics Interface 94*. pp. 83–90. Toronto: Canadian Information Processing Society.

[land95] Landay, James A & Myers, Brad A. 1995. Interactive sketching for the early stages of user interface design. *Human Factors in Computing Systems: "Mosaic of Creativity" CHI '95 Conference Proceedings*. pp. 43–50. New York: ACM.

[lang90] Langston, Peter S. 1990. Little languages for music. *Computing Systems: The Journal of the USENIX Association*. 3(2): 193–287. Berkeley, California: University of California.

[leap97] Leapman, Scott. *Views and Reviews: Graffiti for the Newton—Review #1* [online]. available URL: http://www.amug.org/amug/sigs/newton/nanug/WWWpr18/Reviews9708.html

[leed84] Leedham, C G & Downton, A C & Brooks, C P & Newell, A F. 1984. On-line acquisition of Pitman's handwritten shorthand as a means of rapid data entry. *Human-Computer Interaction: INTERACT '84*. pp. 145–150. Amsterdam: Elsevier Science.

[leed86] Leedham, C G & Downton A C. 1986. On-line recognition of Pitman's handwritten shorthand—An evaluation of potential. *International Journal of Man-Machine Studies*. 24(4): 375–393. London: Academic Press.

[leed94] Leedham, C G. 1994. Historical perspectives of handwriting recognition systems. *IEE Colloquium on 'Handwriting and Pen-Based Input'*. pp. 1/1–3. London: Institute of Electrical Engineers.

[leey94] Lee, Yvonne L. 1994. PDA users can express themselves with Graffiti. *InfoWorld*. 16(40): 30. California: InfoWorld.

[lero94] Leroy, Annick & Müller, Giovanni & Garnett, Guy E. 1994. The design of a pen-based music notation system. *Proceedings of the 20th International Computer Music Conference*, Aarhus, Denmark.

[lime91] Lime music notation software for Apple Macintosh computers. 1991. Products of interest. *Computer Music Journal*. 15(3): 130. Cambridge, Massachusetts: MIT.

[mack94] MacKenzie, I Scott & Nonnecke, Blair & Riddersma, Stan. 1994. Alphanumeric entry on pen-based computers. *International Journal of Human-Computer Studies*. 41(5): 775–792. Salem, Massachusetts: Academic Press.

[mack95] MacKenzie, I Scott. 1995. Input devices and interaction techniques for advanced computing. In *Virtual Environments and Advanced Interface Design*. Barfield, Woodrow & Furness, Thomas A III (eds). pp. 437–470. New York: Oxford Unversity.

[mack97$_a$] MacKenzie, I Scott & Chang, Larry. 1997. *A performance comparison of two handwriting recognizers* [online]. available URL: http:// www.cis.uoguelph.ca/~mac/IWC2.html

[mack97$_b$] MacKenzie, I Scott & Zhang, S. 1997. The immediate usability of Graffiti. *Proceedings of Graphics Interface '97*. pp. 129–137. Toronto: Canadian Information Processing Society.

[mard93] Mardia, K V & Ghali N M & Hainsworth, T J & Howes, M & Sheehy, N. 1993. Techniques for online gesture recognition on workstations. *Image and Vision Computing*. 11(5): 283–294. Guildford, Surrey: Butterworth Scientific.

[maxw84] Maxwell, John Turner III & Ornstein, Severo M. 1984. Mockingbird: A composer's amanuensis. *Byte.* 9(1): 384–401. Peterborough, New Hampshire: McGraw-Hill.

[mcna96$_a$] McNab, Rodger J. 1996. *Interactive Applications of Music Transcription.* MSc thesis. Hamilton, New Zealand: Department of Computer Science, University of Waikato.

[mcna96$_b$] McNab, Rodger J & Smith, Lloyd A & Witten, Ian H. 1996. Signal processing for melody transcription. *ACSC '96: Nineteenth Australian Computer Science Conference.*

[mcqu94] McQueen, Craig & MacKenzie, I Scott & Nonnecke, Blair & Riddersma, Stan & Metz, Malcolm. 1994. A comparison of four methods of numeric entry on pen-based computers. *Proceedings of Graphics Interface '94.* pp. 75–82. Toronto: Canadian Information Processing Society.

[mcqu95] McQueen, J Craig & MacKenzie, I Scott & Zhang, Shawn X. 1995. An extended study of numeric entry on pen-based computers. *Proceedings of Graphics Interface '95.* pp. 215–222. Toronto: Canadian Information Processing Society.

[mere97] Mereu, Stephen W & Kazman, Rick. 1997. Audio enhanced 3d interfaces for visually impaired. *SIGCAPH Newsletter.* vol. 57. pp. 10–15. New York: ACM.

[mess98] Messick, Paul. 1998. *Maximum MIDI: Music Applications in C++.* Greenwich, Connecticut: Manning Publications.

[meye95] Meyer, André. 1995. Pen computing: A technology overview and a vision. *SIGCHI Bulletin.* 27(3): 46–90. New York: ACM.

[meye97] Meyer, André. *PenReport* [online]. Zürich. available URL: http:// www.amug.org/sigs/newton/nanug/PenReport/

[mezi93] Mezick, Dan. 1993. Pen computing catches on. *Byte.* 18(10): 105–112. Peterborough, New Hampshire: McGraw-Hill.

[micr93] Microsoft Press. 1993. *Microsoft Press Computer Dictionary: The Comprehensive Standard for Business, School, Library, and Home.* 2nd edn. Redmond, Washington: Microsoft Press.

[mill$_1$93] Millen, David R. 1993. Pen-based user interfaces. *AT&T Technical Journal.* 72(3): 21–27. New York: AT&T.

[mill$_2$85] Miller, Jim. 1985. Personal Composer. *Computer Music Journal.* 9(4): 27–37. Cambridge, Massachusetts: MIT.

[mora95] Moran, Thomas P & Chiu, Patrick & van Melle, William & Kurtenbach, Gordon. 1995. Implicit structures for pen-based systems within a freeform interaction paradigm. *Human Factors in Computing Systems: "Mosaic of Creativity" CHI '95 Conference Proceedings.* pp. 487–494. New York: ACM.

[mosa94] *NewsRelease: MIT Introduces Four MOSAIC Character Generator Packages* [online]. 1994. American-Tele-Systems. available URL: http://www.amtelcom.com/PressReleases/mit1294.html

[mosa95$_a$] *Mosaic Input Technologies* [online]. American-Tele-Systems. available URL: http://www.amtelcom.com/Mosaic/

[mosa95$_b$] Mosaic Input Technologies. 1995. *Gesture Mosaic User Guide* [online]. American-Tele-Systems. available URL: http://www.amtelcom.com/Mosaic/userguide.html

[mosa95$_c$] *NewsRelease: MIT Announces Gesture Mosaic—A New Pen-Based Entry Software for the Apple Newton* [online]. 1995. American-Tele-Systems. available URL: http://www.amtelcom.com/PressReleases/mit495.html

[mosa95$_d$] Newton software review: Gesture Mosaic—Better than Graffiti? [online]. 1995. *Pen Computing Issue 6.* Pen Computing Magazine. available URL: http://pencomputing.com/PCM_6/review_gesture_mosaic.html

[mosa96] *NewsRelease: Mosaic Input Technologies releases freeware version of Gesture Mosaic* [online]. 1996. American-Tele-Systems. available URL: http://www.amtelcom.com/PressReleases/mit696.html

[mosa97] *Gesture Mosaic* [online]. American-Tele-Systems. available URL: http://www.amtelcom.com/Mosaic/gmgif/quickref.gif

[moto96] *QuickPrint for Embedded Systems* [online]. 1996. Motorola. available URL: http://www.mot.com/MIMS/lexicus/products/QuickPrint/Info.html

[myer85] Myers, B. 1985. The importance of percent-done progress indicators for computer-human interfaces. *Human Factors in Computing Systems: CHI '85 Conference Proceedings*. pp. 11–17. New York: ACM.

[myna94$_a$] Mynatt, Elizabeth D. 1994. Auditory presentation of graphical user interfaces. *Auditory Display: Sonification, Audification, and Auditory Interfaces*. Kramer, Gregory (ed.). pp. 533–555. Reading, Massachusetts: Addison-Wesley.

[myna94$_b$] Mynatt, Elizabeth D & Weber, Gerhard. 1994. Nonvisual presentation of graphical user interfaces: Contrasting two approaches. *Human Factors in Computing Systems: "Celebrating Interdependence" CHI '94 Conference Proceedings*. pp. 166–172. New York: ACM.

[niel92] Nielson, Jakob. 1992. Finding usability problems through heuristic evaluation. *Human Factors in Computing Systems: "Striking a Balance" CHI '92 Conference Proceedings*. pp. 373–380. New York: ACM.

[niel93] Nielson, Jakob. 1993. *Usability Engineering*. Boston: Academic Press.

[niff95] NIFF Technical Committee. 1995. *Notation Interchange File Format 6a* [online]. Grande, Cindy, NIFF Technical Coordinator. email: 72723.1272@compuserve.com available URL: http://www.jtauber.com/music/encoding/niff/spec/

[nola95] Nolan, Ron S & Nolan, Susan A. 1995. *Computer Music: An Interactive Documentary* [CD-ROM]. Cyberlearning Collection. California: University of California.

[norm$_1$88] Norman, Donald A. 1988. *The Psychology of Everyday Things*. New York: BasicBooks.

[norm$_2$86] Norman, Kent L & Weldon, Linda J & Shneiderman, Ben. 1986. Cognitive layouts of windows and multiple screens for user interfaces. *International Journal of Man-Machine Studies.* 25(2): 229–248. London: Academic Press.

[obor95] Oborne, David J. 1995. *Ergonomics at Work: Human Factors in Design and Development.* 3rd edn. Chichester, West Sussex, England: Wiley.

[ouel95] Ouellette, Daniel. 1995. *The Pen Connection: A Guide to Pen Computing.* New York: McGraw-Hill.

[over96] Overton, Randall C & Taylor, L Rogers & Zicker, Michael J & Harms, Harvey J. 1996. The pen-based computer as an alternative platform for test administration. *Personnel Psychology: A Journal of Applied Research.* 49(2): 455–464. Washington, District of Columbia: Personnel Psychology.

[past83] Pastoor, Siegmund & Schwarz, Elmar & Beldie, Ion P. 1983. The relative suitability of four dot-matrix sizes for text presentation on color television screens. *Human Factors.* 25(3): 265–272. Santa Monica, California: The Human Factors Society.

[pede95] Pedersen, Elin Rønby & McCall, Kim & Moran, Thomas P & Halasz, Frank G. 1995. Tivoli: An electronic whiteboard for informal workgroup meetings. In *Readings in Human-Computer Interaction: Toward the Year 2000.* Baecker, Ronald & Grudin, Jonathan & Buxton, William A S & Greenberg, Saul (eds). pp. 509–516. San Francisco, California: Morgan Kaufmann.

[pere96] Pérez-Quiñones, Manuel A & Sibert, John L. 1996. A collaborative model of feedback in human-computer interaction. *Human Factors in Computing Systems: "Common Ground" CHI '96 Conference Proceedings.* pp. 316–323. New York: ACM.

[perk95] Perkins, Ronald & Blatt, Louis A & Workman, Daniel & Ehrlich, Susan F. 1995. Iterative tutorial design in the product development cycle. In *Readings in Human-Computer Interaction: Toward the Year 2000.* Baecker, Ronald & Grudin, Jonathan & Buxton, William A S & Greenberg, Saul (eds). pp. 881–885. San Francisco, California: Morgan Kaufmann.

[pick97] Picking, Richard. 1997. Reading music from screens vs paper. *Behaviour & Information Technology*. 16(2): 72-78. London: Taylor & Francis.

[pisz77] Piszczalski, Martin & Galler, Bernard A. 1977. Automatic music transcription. *Computer Music Journal.* 1(4): 24-31. Menlo Park, California: People's Computer Company.

[pisz79] Piszczalski, Martin & Galler, Bernard A. 1979. Computer analysis and transcription of performed music: A project report. *Computers and the Humanities.* 13(3): 195-206. New York: Pergamon Press.

[pitc94] Pitchforth, Mathew. 1994. *Hand Written Music Recognition.* Honours project. Palmerston North, New Zealand: Department of Computer Science, Massey University.

[prin97] Prince, Robert. *Views and Reviews: Graffiti for the Newton—Review #2* [online]. available URL: http://www.amug.org/amug/sigs/newton/ nanug/WWWpr18/ Reviews9480.html

[quic96] Magic Cap software review: Lexicus Quickprint [online]. 1996. *Pen Computing Issue 8.* Pen Computing Magazine. available URL: http:// www.pencomputing.com/PCM8/ P8_lexicus_review.html

[rade96] Rader, Gary M. 1996. Creating printed music automatically. *Computer.* 29(6): 61–68. Long Beach, California: IEEE Computer Society.

[read69] Read, Gardner. 1969. *Music Notation: A Manual of Modern Practice.* London: Victor Gollanz.

[road81] Roads, C. 1981. A note on music printing by computer. *Computer Music Journal.* 5(3): 57–59. Cambridge, Massachusetts: MIT.

[ross70] Ross, Ted. 1970. *The Art of Music Engraving and Processing: A Complete Manual Reference and Text Book on Preparing Music for Reproduction and Print.* Miami: Hansen Books.

[roth89$_a$] Rothstein, Joseph. 1989. Grandmaster MusicEase notation software. *Computer Music Journal.* 13(2): 94–95. Cambridge, Massachusetts: MIT.

[roth89$_b$] Rothstein, Joseph. 1989. MusicEase notation software for IBM PCs. *Computer Music Journal.* 13(3): 105–106. Cambridge, Massachusetts: MIT.

[roth91] Rothstein, Joseph. 1991. Coda Finale-PC 2.0 notation software for IBM PCs. *Computer Music Journal.* 15(4): 115–118. Cambridge, Massachusetts: MIT.

[rubi91] Rubine, Dean. 1991. Specifying gestures by example. *Computer Graphics.* 25(4): 329–337. New York: ACM SIGGRAPH.

[ryan91] Ryan, Bob. 1991. Dynabook Revisited with Alan Kay. *Byte.* 16(2): 203–208. Peterborough, New Hampshire: McGraw-Hill.

[scar96] Scariot, Brenda. 1996. *Lexicus, a Division of Motorola, Introduces Handwriting Recognition Software for Magic Cap Communicators: Lexicus QuickPrint Enables Wireless Communicators to Recognize Natural Handwriting* [online]. Motorola. available URL: http:// www.mot.com/MIMS/ lexicus/press/releases/Jul_31_95.html

[schw83] Schwarz, Elmar & Beldie, Ion P & Pastoor, Siegmund. 1983. A comparison of paging and scrolling for changing screen contents by inexperienced users. *Human Factors.* 25(3): 279–282. Santa Monica, California: The Human Factors Society.

[self94] Selfridge-Field, Eleanor & Correia, Edmund, Jr. 1994. *Musical Information in Musicology and Desktop Publishing.* Menlo Park, California: Center for Computer-Assisted Research in the Humanities.

[shne92] Shneiderman, Ben. 1992. *Designing the User Interface: Strategies for Effective Human-Computer Interaction.* 2nd edn. Reading, Massachusetts: Addison-Wesley.

[sign91] Signell, Karl. 1991. Music notation software. *Journal of the American Musicological Society.* 44(1): 136–148. Richmond, Virginia: American Musicological Society.

[snyd79] Snyder, Harry L. 1979. The sensitivity of response measures of alphanumeric legibility to variations in dot matrix display parameters. *Human Factors.* 21(4): 457–471. Santa Monica, California: The Human Factors Society.

[stew94] Stewart, Alsop. 1994. Innovative Graffiti might actually make PDAs usable. *InfoWorld.* 16(39): 130. California: InfoWorld.

[styn90] Styne, Bruce A. 1990. Command history in a reversible painting system. *Computer Animation '90.* pp. 149–164. Tokyo: Springer-Verlag.

[tapi95] Tapia, Mark A & Kurtenbach, Gordon. 1995. Some design refinements and principles on the appearance and behavior of marking menus. *UIST '95 User Interface and Software Technology: Proceedings of the Eight ACM Symposium.* pp. 189–195. New York: ACM.

[tapp90] Tappert, Charles C & Suen, Ching Y & Wakahara, Toru. 1990. The state of the art in on-line handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 12(8): 787–808. New York: IEEE Computer Society.

[tapp91] Tappert, C C. 1991. Speed, accuracy, and flexibility trade-offs in on-line character recognition. In *Character and Handwriting Recognition: Expanding Frontiers.* Wang, P S P (ed.). World Scientific Series in Computer Science. vol. 30. pp. 79–95. Singapore: World Scientific Publishing.

[taup95] Taupin, Daniel. 1995. *MusicTEX: Using TEX to write Polyphonic or Instrumental Music.*

[tayl92] Taylor, Eric. 1992 (1991). *The AB Guide to Music Theory.* pt 2. London: The Associated Board of the Royal Schools of Music.

[tayl94] Taylor, Eric. 1994 (1989). *The AB Guide to Music Theory.* pt 1. London: The Associated Board of the Royal Schools of Music.

[thim90] Thimbleby, Harold. 1990. Undo. ch. 12. *User Interface Design.* pp. 261–286. New York: ACM.

[trau91] Traux, Barry. 1991. Capturing musical knowledge in software systems. *Interface*. 20(3–4): 217–233. Lisse: Swets & Zeitlinger.

[unis97] *Unistroke Alphabet* [online]. available URL: ftp://parcftp.xerox.com/pub/unistrokes/mnemonic.ps

[whal96] Whaley, Dave. 1996. *Motorola Lexicus Announces Handwriting Recognition on New Low-Power Microprocessor: DragonBall and QuickPrint Deliver High Performance in a Small Memory Footprint.* Motorola. available URL: http://www.mot.com/MIMS/lexicus/press/releases/Sept_25_95.html

[whee96] Wheelwright, Geof. 1996. Handwriting recognition comes in from the cold: Graffiti leads the second wave of pen-enhanced data-entry technology [online]. *The Computer Paper Online*. Canada Computer Paper. available URL: http://www.tcp.ca/1996/Mar96/Portable/Handwrit/Handwrit.html

[wigg93] Wiggins, Geraint & Miranda, Eduardo & Smaill, Alan & Harris, Mitch. 1993. A framework for the evaluation of music representation systems. *Computer Music Journal.* 17(3): 31–42. Cambridge, Massachusetts: MIT.

[wolf86] Wolf, Catherine G. 1986. Can people use gesture commands? *SIGCHI Bulletin*. 18(2): 73–74. New York: ACM.

[wolf87] Wolf, Catherine G & Morrel-Samuels, Palmer. 1987. The use of hand-drawn gestures for text editing. *International Journal of Man-Machine Studies*. 27(1): 91–102. London: Academic Press.

[yave85] Yavelow, Christopher. 1985. Music software for the Apple Macintosh. *Computer Music Journal.* 9(3): 52–67. Cambridge, Massachusetts: MIT.

[yave87] Yavelow, Christopher. 1987. A report on the workshop for music notation by computer. *Computer Music Journal.* 11(2): 65–70. Cambridge, Massachusetts: MIT.